SegmentPerturb: Effective Black-Box Hidden Voice Attack on Commercial ASR Systems via Selective Deletion

by

Ganyu Wang

A thesis submitted to the
School of Graduate and Postdoctoral Studies in partial
fulfillment of the requirements for the degree of

Master of Science in Computer Science

Faculty of Science

University of Ontario Institute of Technology (Ontario Tech University)

Oshawa, Ontario, Canada

August 2021

THESIS EXAMINATION INFORMATION

Submitted by: **Ganyu Wang**

Master of Science in **Computer Science**

Thesis title: SegmentPerturb: Effective Black-Box Hidden Voice Attack on Commercial ASR Systems via Selective Deletion

An oral defense of this thesis took place on July 27, 2021 in front of the following examining committee:

Examining Committee

Chair of Examining Committee	Dr. Patrick Hung
Research Supervisor	Dr. Miguel Vargas Martin
Supervisory Committee	Dr. Bill Kapralos
External Examiner	Dr. Faisal Qureshi

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

Voice control systems continue becoming more pervasive as they are deployed in mobile phones, smart home devices, automobiles, etc. Commonly, voice control systems have high privileges on the device, such as making a call or placing an order. However, at the same time, they are vulnerable to voice attacks, which may lead to serious consequences. In this thesis, SegmentPerturb was proposed to craft hidden voice commands via inquiring the target models. The basic idea of SegmentPerturb is that the original command audio was separated into multiple segments and a certain degree of perturbation was applied to each segment by probing the target speech recognition system. Experiments were conducted on four popular commercial speech recognition APIs plus one smart home device to show the practicability of SegmentPerturb. Results suggest that SegmentPerturb can generate voice commands which can be recognized by the machine but are hard to understand by a human.

Keywords— Automatic Speech Recognition System; Hidden Voice Command

Author's Declaration

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

The research work in this thesis that was performed in compliance with the regulations of Research Ethics Board under REB 16091 certificate file number.

_____ Ganyu Wang

Statement of Contributions

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication. I have used standard referencing practices to acknowledge ideas, research techniques, or other materials that belong to others. Furthermore, I hereby certify that I am the sole source of the creative works and/or inventive knowledge described in this thesis.

Acknowledgements

We are grateful for the support from the Natural Sciences and Engineering Research Council of Canada.

Contents

List of Figures

List of Tables

List of Abbreviations

ASR Automatic Speech Recognition.

HMMs Hidden Markov Models.

GMMs Gaussian Mixture Models.

DNNs Deep Neural Networks.

LSTM Long-Short Term Memory.

DBLSTM Deep Bidirectional Long-Short Term Memory.

AMP Amplifier.

LPF Low-pass filter.

ADC Analog-to-Digital Converter.

FFT Fast Fourier Transform.

DFT Discrete Fourier Transform.

MFC Mel-frequency Cepstrum.

MFCCs Mel-Frequency Cepstral Coefficients.

CTC Connectionist Temporal Classification.

MIPF Monotonically Increasing Perturbation Function.

APR Average Perturbation Rate.

STFT Short-Time Fourier Transform.

List of Symbols

x A waveform.

x_0 The original waveform of the command.

x^* The perturbed waveform.

$f_m(\cdot)$ The ASR system as a function.

$f_h(\cdot)$ The human comprehension as a function.

y_m The transcription result of the command by the ASR.

y_h The transcription result of the command by the human comprehension.

y^* The transcription result of the perturbed waveform x^*. ($f_m(x^*)$).

δ The perturbation added into the waveform x.

$S(x_0)$ The set of the correct transcription of the input waveform x_0.

Chapter 1

Introduction

Voice control systems are becoming increasingly popular and pervasive. For example, the well-known Siri from Apple, Cortana from Microsoft, iFLYTEK speech input and Baidu speech input are widely deployed in devices used in our daily lives. In 2020, the estimated population in the US who use a voice assistant at least monthly was 128 million, representing 44.2% of internet users and 38.5% of the total population [6]. Particularly, for smart home devices such as Amazon Alexa, Google Home, etc. voice interface is the only interactive method for the users. Voice control is becoming one of the mainstream human-computer interaction modalities.

Attacking a voice assistant has a high return and a low risk for attackers. First, voice assistants always have a high priority in controlling the device, which means that compromising a voice assistant will give the attackers high-level control on the victim's devices. It is very common that voice assistants control all of the smart electric appliances in the facilities. Furthermore, payment can be made by a voice assistant. Setting up payment with a voice assistant is even being advocated in the advertisement of the leading banks. Second, most of the voice control devices solely use conversation as the interactive method, the user cannot check what instructions the voice control devices have executed, which provides stealthiness for the attackers. An example for the real-world application could

be that the attacker embeds the malicious command into the sound of a famous YouTube video with a large number of watches. When the hosts play the video with their speakers on, the audio can be recognized as meaningful commands by their smart home device and there is little chance that the users are aware of this attack. Therefore, security should be considered an important issue for the design of voice assistants.

However, the key technology of voice assistants, Automatic Speech Recognition (ASR), is highly vulnerable to a variety of attacks. Previous works have proven that the adversary can generate an audio sample that can be recognized as a meaningful command by the ASR, but at the same time being very hard for humans to understand [8,13,37]. This kind of attack is called a hidden voice attack. Another kind of voice attack is called adversarial attacks, whereby the adversary tries to add the smallest amount of perturbation to the original audio, but making the ASR recognize it as another command which is totally different from the original command (adversarial attack on voice interface) [8, 14, 31, 41]. Other attacks take advantage of the property of the hardware, using ultrasonic waves to inject inaudible command on the ASR [40, 42]. If the adversary deliberately designs the audio input, it is feasible for them to inject malicious commands on the voice control system.

This research studied the hidden voice attack on ASR, to help enhance the security of the voice control system. Previous research on hidden voice attacks is not practical enough for a real-world attack. The most stable attack on voice assistant is the hardware attack [35, 42], but those attacks normally need bulky instruments and close contact with the victim device. It works well for attacking a specific nearby target, but it does not work well for long-range and wide-range attacks. Apart from hardware attacks, hidden voice attacks and adversarial attacks both use the speaker to attack the voice assistant, implying that they can be applied to long-range attacks as long as the adversaries compromise the speakers of the victim. However, to date, the research on voice attack is still not practical in the real world, the attack distance is limited below two meters and only a few attacks [8,13,17,41] work for the commercial ASRs. In this thesis, an attack called SegmentPerturb was proposed to perform a hidden voice attack on the ASR. This attack achieves an equivalent attack performance

as the state-of-the-art voice attack [8, 17].

Besides, SegmentPerturb does not make any assumptions as to what feature extraction method the ASR system uses. Nowadays, modern ASR is involved in rapid development; some are using classical signal processing methods, while others are moving towards end-to-end methods with the simplest feature extraction method. For example, just use the waveform as the feature. Therefore, the spectrum analysis assumptions on the ASR may fail in the future. SegmentPerturb does not make an assumption on the ASR and the general idea is that the adversaries probe the ASR to selectively reduce a certain amount of information for each segment of the audio command. Because of the high noise resistance of the ASR and the redundancy of information in speech audio, a large amount of information can be removed from the audio command and the command can still be recognized by the ASR.

The contributions of our work are as follows.

- **SegmentPerturb.** A perturbation method to perform a hidden voice attack on any modern ASR system via inquiring is proposed. The idea of SegmentPerturb is to separate the audio into multiple equal-length segments and probe the ASR to distort the unimportant segments. Comparing with the similar attacks, our attack produces equivalent or more unintelligibility in the attack audio samples and has a longer attack distance, using a simpler and more straightforward attack method.

- **SegmentPerturb further relaxes the assumptions on the feature extraction method of the ASR for black-box attacks.** Previous works [8, 13, 37] on black-box hidden voice attack models have some assumptions on the signal processing phase of the ASR. But SegmentPerturb is based on the universal property which exists in all of the ASRs and even in human comprehension, that if the audio is separated into multiple equal-length segments, the importance of every segment is different. The ASR may pay more attention to some segments while somewhat ignoring others. Besides, given the high noise resistance of modern ASRs, a large amount of perturbation

can be added into almost every segment in one audio command.

- **Full Study on Segmented Perturbation (SegmentPerturb).** A full study was conducted on SegmentPerturb using four of the most popular speech recognition APIs in the market (Google speech recognition, Wit.ai speech API , IBM speech-to-text, Azure speech services). Besides, a real scenario attack using SegmentPerturb was demonstrated on Google Home. A real scenario of a bedroom was set up and the Google Home was placed at different positions. The attack samples of SegmentPerturb can be recognized by the Google Home at most of the positions. The effectiveness of SegmentPerturb is studied and verified. SegmentPerturb can significantly reduce the intelligibility of the command while maintaining the intelligibility to the machine.

- **Perturbation Framework**. A general framework to craft hidden voice commands by gradually reducing the human intelligibility of an audio command was formally defined. The basic idea of the framework is to use a function that can increase the level of perturbation to the audio while iteratively inquiring the ASR to add a certain amount of perturbation while preserving its intelligibility by the ASR.

The rest of the thesis is organized as follows: the second chapter introduces the background for speech recognition system and related attack on speech recognition system; the third chapter illustrates the method for performing the attack and the intuition behind it; the fourth chapter expands on the experiment for testing the performance of the attack; the fifth chapter presents a detailed comparison with the state-of-the-art works and discusses the property of SegmentPerturb; the sixth chapter summarizes limitations, future work of SegmentPerturb, and concludes the thesis.

Chapter 2

Related Works

2.1 Technical Background: Speech Recognition

The technical background of speech recognition is illustrated in this section. The structure of the ASR is first introduced, followed by the details of each component in it.

Before deep neural networks were introduced into ASR, most ASRs were Hidden Markov Models (HMMs) [19,25], and Gaussian Mixture Models (GMMs). Later, as the strong learning ability of deep learning was discovered, researchers started using Deep Neural Networks (DNNs) to substitute some procedures in the traditional models, which is called a hybrid method. In 2013, Graves et al. [21, 22] used Deep Bidirectional Long-Short Term Memory (DBLSTM) to improve the accuracy in speech recognition. In 2014 their research moved to an end-to-end speech recognition system using Long-Short Term Memory (LSTM) [20]. And recent researches mostly used the end-to-end method to improve the accuracy of speech recognition. Zhang et al. [43] trained a very deep neural network with 15 layers to build the end-to-end ASR system, achieving a 10.5% word error rate without using any dictionary or language model. Chiu et al. [18] built a unidirectional LSTM encoder for stream recognition and achieved 5.6% word error rate.

Most of the Tech giants promoted the study of speech recognition and even provided

commercial speech recognition services. IBM Watson Research Center proposed their conversational speech recognition system in [32] and they also provided the commercial speech recognition API service [3]. Baidu Research proposed Deep Speech [23] which is an end-to-end ASR model using three fully connected layers, one bidirectional recurrent layer, and one standard softmax layer. They achieved a 16% error rate on the full test set. Microsoft Research's conversational speech recognition system had achieved human parity [38, 39], achieving a 5.9% word error rate. Microsoft [5] and Google [1] also provided their commercial speech recognition service. Most of the commercial speech recognition services used a neural network model to achieve a low word error rate.

2.1.1 Structure of Modern ASR

The structure of the modern ASR is depicted in Fig. 2.1. This thesis only focuses on the modern ASRs which use DNN instead of HMM. The speech recognition pipeline includes the following phase: the sound is captured by the sound input device (hardware), then the signal is passed to the feature extraction part (signal processing), the features are input into the DNN model for model inference and to obtain the probability matrix for each time frame. The probability matrix is then decoded using a decoding algorithm; some ASRs may also apply a language model to correct the output label.

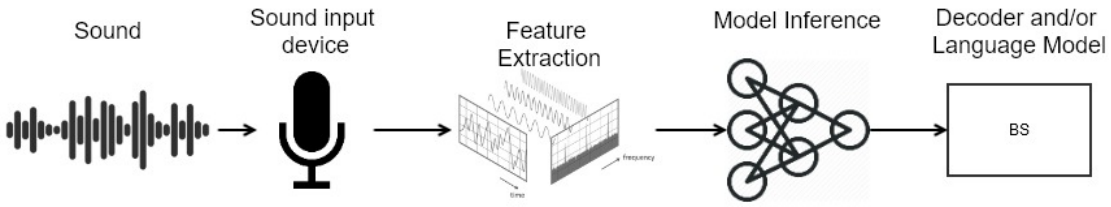


Figure 2.1: The structure of modern ASRs.

Sound Input Device The sound input device is the microphone hardware that captures the vibrations in the air and converts them into an electronic signal. The most common

microphones in the market include dynamic microphone and condenser microphone. The dynamic microphone uses a coil of wire and a magnetic to capture the vibration in the air, the vibration is turned into electromotive force according to the Faraday law of electromagnetic induction. The condenser microphone uses the changing of the distance of the diaphragm from the other plate to capture the vibration in the air. The distance of the two plates influences the capacitor, and thus turns the vibration into an electronic signal. The structure of the sound input device is depicted in Fig. 2.2. When the vibration in the air is captured by the mechanical structure of the vibration, the signal is passed to an Amplifier (AMP) to increase the amplitude of the signal. The sound signal is passed through a Low-pass filter (LPF) to filter out the high-frequency noise. This signal is converted into the digital signal using a Analog-to-Digital Converter (ADC).

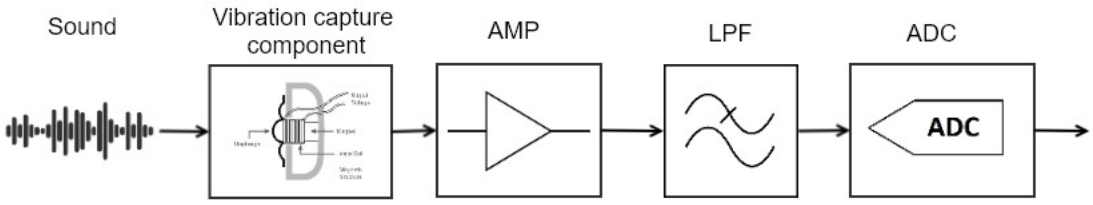


Figure 2.2: The structure of the sound input device.

Feature Extraction The audio signal is first separated into short time frames, the length of the time frame varies from 20 ms to 40 ms. Suppose each time frame has N sample points in it, the value of the i^{th} sample point is x_i. We denote the sequence of sample point in one time frame as $\{x_k\} := x_0, x_1, ..., x_{N-1}$.

The most commonly used method for feature extraction in speech recognition is Fast Fourier Transform (FFT) and Mel-frequency Cepstrum (MFC). But with the rapid development of deep learning, some end-to-end models for ASR make the original time frame of the sequence as one optional output for the feature extraction phase.

a) FFT: The FFT is a way to compute the Discrete Fourier Transform (DFT) of a

given sequence. It transforms the representation of the time domain signal into the frequency domain.

The DFT can be represented by Eq. 2.1. The DFT of the given sequence $\{x_n\}$ of one time frame of the audio signal is $\{X_k\} := X_0, X_1, ..., X_{N-1}$.

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{-i2\pi}{N}kn} \tag{2.1}$$

The power spectrum sequence $\{p_k\}$ of the signal $\{X_k\}$ is shown in Eq. 2.2, which is the square of the absolute of the DFT of the signal.

$$p_k = |X_k|^2 \tag{2.2}$$

b) MFC: The MFC is better at representing human audition. This is because humans are better at recognizing the difference in the lower frequency band and the Mel-scale represents these characteristics better than other feature recognition methods. To get the Mel-Frequency Cepstral Coefficients (MFCCs) of a given time frame, it takes the following steps.

First, perform a DFT on the signal using the FFT Eq. 2.1 and 2.2 to obtain the spectrum of the Fourier transform in the format of $\{p_k\}$. Then the frequency of the $\{p_k\}$ is mapped to the Mel scale using Eq. 2.3

$$m_k = 2595 \log_{10}(1 + \frac{f_k}{700}) \tag{2.3}$$

Third, take the logs of the power at each of the Mel frequencies. The $\{L_n\}$ to be derived from this part can be represented by Eq. 2.4:

$$L_k = \log_{10}(|X_k|^2) \tag{2.4}$$

Fourth, take the discrete cosine transform of the Mel log power, regarding it as a signal. The discrete cosine transform of the signal for the Mel log power is obtained using Eq.2.5.

$$F_k = \sum_{n=0}^{N-1} L_n \cos\left(\frac{(2n+1)k\pi}{2N}\right) \tag{2.5}$$

c) Simplified: with the development of deep learning, some end-to-end speech recognition systems treats simplified feature extraction as an option. The feature extraction will only include separating the audio into small time frames. The time frames will be used as the features to be inputted in the deep learning model.

Model Inference The features for each time frame are passed into the DNN model for inference. The input of the model is a sequence of features for the time frames, denoted as $\{x_i\} := x_1, x_2, ...x_{N-1}$. For the time frame at position i, the model will output the probability matrix for each character $P\{c_k|x_i\}$, where $c_k \in \{a, b, c, ...z, space, apostrophe, hyphen, blank\}$. *space*, *apostrophe* and *hyphen* are the corresponding characters $\{' ', ''', '-'\}$. *blank* is the special character used in Connectionist Temporal Classification (CTC) to represent the null.

a) Convolutional Neural Network (CNN): CNN is a kind of deep neural network which uses convolutional layers as components. The convolution layer learns the spatial characteristics from the previous layers. Therefore it is commonly used in image learning. The convolution in deep learning is an operation that use a convolutional kernel to scan through the input to get an output matrix, for each step, the corresponding elements in the input matrix was multiplied by the kernel to get a convolution output.

Song et al. [34] and Chang et al.'s [15] work on building end-to-end ASR applied Convolutional layers in their network. But convolutional layers do not fit the speech recognition problem very well, because the convolutional transform of the features does not well represent the characteristics of sound. Intuitively, there is no strong spatial relation in the features of the sound in one time frame. Besides, CNN normally takes fixed length and width input, but the characteristic of sound is that it has a

variable length. Therefore, all those works used recurrent layers at the output end of the model following the convolutional layers.

b) Recurrent Neural Network (RNN): RNNs can deal with the problem of variable length data that CNNs and perceptrons are not capable of. Therefore, RNNs can be applied to learning from sequence data such as sound or video. Naturally, it is also used in speech recognition [22, 29].

An RNN takes the input with the length of L and it will generate the output of the same length of L corresponding to the data at each position of the sequence. Consider an RNN with the input x_t and the previous state h_{t-1}. The graph for the structure of a normal RNN is shown in Fig. 2.3. The RNN will output the current state h_t. A common data processing of RNN can be represented by Eq. 2.6 and 2.7:

$$S_t = W_x \cdot x_t + W_h \cdot h_{t-1} \tag{2.6}$$

$$h_t = \tanh(S_t) \tag{2.7}$$

In particular, an RNN can be used to deal with the context data, which means that the data at one position in the sequence has some connection with the data at another position of the sequence. For example, in the speech recognition inference model, if the current position has a high probability of outputting "r" which is a consonant, the output at the position nearby would be more likely to be a vowel. Since speech is a kind of data that is temporally correlated, an RNN has a great advantage in solving the speech recognition problem.

c) Long Short-Term Memory (LSTM): Practically, it is hard for an RNN to find the connection between the data at the current position and the position which is far away from it, which is called the long-term dependencies problem. Therefore, the idea of LSTM was proposed to address the long-term dependencies problem [24]. The basic idea is applying a "forget gate" and an "input gate" to modify the state C_t

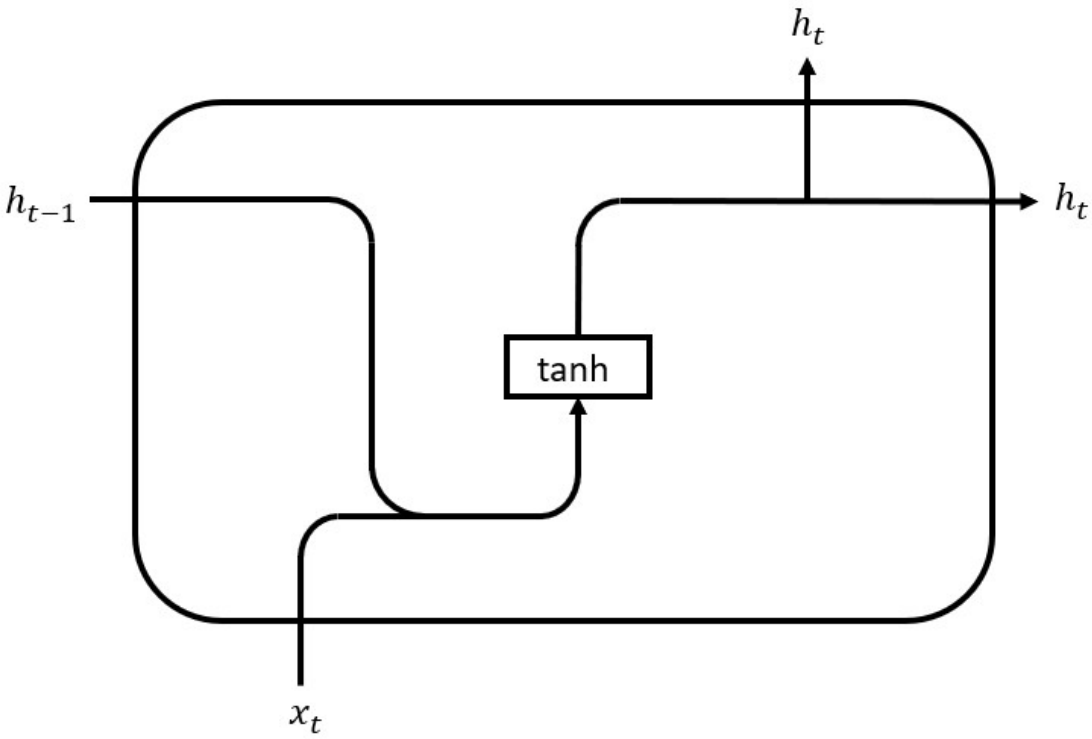


Figure 2.3: The structure of an RNN.

of the network, and the output h_t is derived from the current state C_t, current input x_t and the previous output h_{t-1}. The structure of the LSTM is depicted in Fig. 2.4

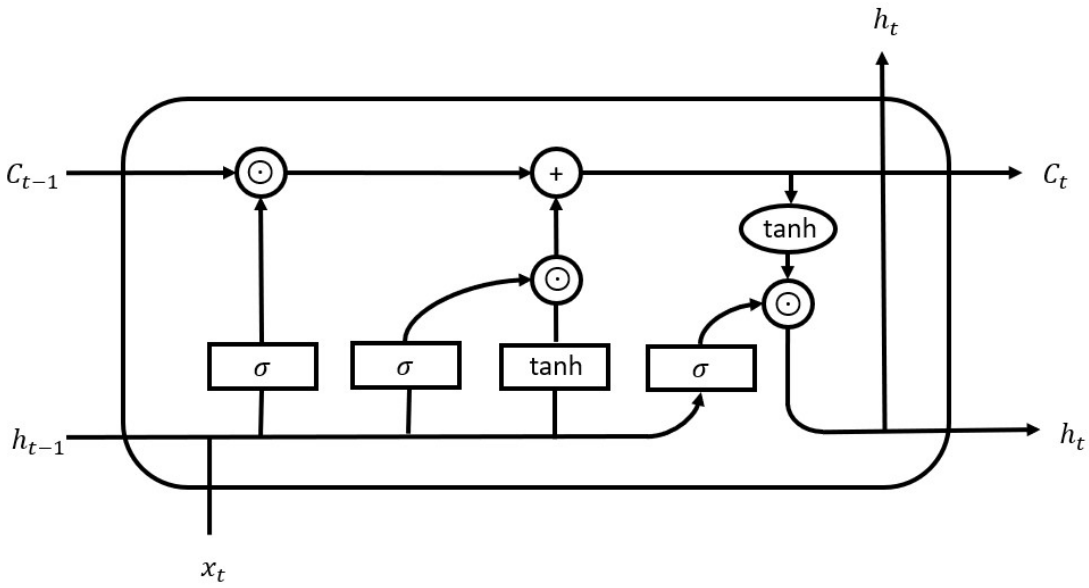


Figure 2.4: The structure of an LSTM.

The "forget gate" is represented in Eq. 2.8, in which W_f and b_f are the network weight and bias, $\sigma(\cdot)$ is the sigmoid function. $[h_{t-1}, x_t]$ is the affine transformation $[h_{t-1}, x_t] = W_h \cdot h_{t-1} + W_x \cdot x_t + b$.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.8}$$

The "input gate" decides which element in C_{t-1} should be updated. It is defined by the following formulas.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.9}$$

The state of the current input C_t is calculated using the *tanh*.

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.10}$$

Combining the "input gate" and the current state of input x_t, this value can be used to update the current state of the LSTM C_t.

Then, the current state C_t of the LSTM is updated by the "forget gate", "input gate", and the state of the current input. The formula is shown below. The $\odot$ is the element-wise multiplication.

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{2.11}$$

The output of the LSTM is the combination of the input state x_t, h_t and the current state C_t. The input state is passed through a sigmoid network to determine which part should be output in the current state. The current state C_t is passed through a tanh to compress the value to the segment of $(-1, 1)$. This procedure can be represented in the formulas below.

$$o_t = \sigma(W_0 \cdot [h_{t-1}]) \tag{2.12}$$

$$h_t = o_t \odot \tanh(C_t) \tag{2.13}$$

Since LSTMs are capable of learning long-term connections in the sequence data, they are widely used in dealing with sequence data. And it is proven feasible and efficient for speech recognition [18, 20, 22, 23].

Decoder The last step of speech recognition is the decoding part. From the previous step, the deep learning model outputs a probability matrix for the input sequence. The first dimension of the probability matrix is corresponding to the time frame of the input sound. The length of the probability matrix is the same as the length of the time frame sequence. The second dimension of the probability matrix corresponds to the character in the alphabet. The element A_{ik} in the probability matrix is the probability of the character c_k at the time frame x_i.

In speech recognition, many possible transcriptions can be applied to a probability

matrix. There are a lot of words in English that have the same or similar pronunciation and each person may have an accent. To determine which characters (or words) could be the best one in the context, a decoder was used to get the most possible characters from the given probability matrix. The most commonly used decoding algorithm for speech recognition is the beam search, which can determine the output sequences with a high probability. Beam search is a heuristic searching algorithm, which can largely reduce the searching space, and it has been proved feasible and efficient for the speech recognition task.

2.2 Attack on ASR

2.2.1 Machine Learning Model Level Voice Attacks

Hidden Voice Command. Hidden voice command is a kind of attack on ASR in which the adversary adds the largest perturbation on the audio, which totally destroys the intelligibility of the audio but the audio can still be recognized as a meaningful command by the ASR system. Carlini et al. [13] analyzed the hidden voice command in white-box and black-box settings. Under the white-box setting, they chose the CMU sphinx [27] (traditional speech recognition model) as the target model and generated hidden voice commands which cannot be understood by humans but can be understood by ASR. Under a black-box setting, they can generate noise audio that can be recognized by machines but is hard to be understood by humans. Abdullah et al. [8] designed a practical hidden voice attack that attacked the signal processing phase of the ASR system. They added four types of perturbation (time domain inversion, random phase generation, high-frequency addition, time scaling) to the audio samples. Each kind of perturbation had some parameters and they generated a set of adversarial samples with different perturbation parameters and fed them into the ASR. They discarded the audio samples which cannot be transcribed correctly by the ASR and choose the audio file with the best parameter. Chen et al. [16] studied how to improve over-the-air voice attacks in a white-box setting, attacking the speech recognition

model Deep Speech [23] from the Baidu Research team. They found that the reason for the difficulty of an over-the-air attack is the frequency-selectivity effect caused by the devices and channel, and they designed their over-the-air attack which achieved a 90% success rate over a distance of 6 m.

Adversarial Attack Against ASR. In an adversarial attack against the ASR system, the adversary adds the least possible amount of perturbation on the audio to make the machine recognize the audio as another transcription that is different from the original one. The adversarial attack on neural networks is first proposed by Szegedy et al. [36] in the computer vision domain.

Yuan et al. [41] performed the adversarial attack in a white-box setting and embedded the malicious command into the song. The song they generated can be recognized as malicious command by the voice control devices. They performed their attack on the popular open-source speech recognition platform Kaldi [4,30], which is a GMM-HMM speech recognition platform. They used gradient descent to craft the audio to approximate the same probability matrix of the audio of a meaningful command. Schönherr et al. [33] crafted adversarial attack samples in a white-box setting on the Kaldi platform [4] using a generative system based on neural networks. They were more inclined to add a component that is lower than the hearing threshold to prevent the human from perceiving the change. Qin et al. [31] further improved Schönherr et al.'s [33] work, using a similar psychoacoustic hiding method (applying the hearing threshold to perturbation adding) but attacked a modern Lingvo ASR system. They also made the adversarial samples more robust, having a better success probability for the over-the-air attack.

Kwon et al. [26] proposed a new kind of adversarial attack on ASR called selective audio adversarial samples. The selective audio adversarial sample is misclassified as the targeted phase by the victim ASR but is correctly classified by the protected ASR. Most of the previous adversarial attacks were based on white-box since black-box adversarial was difficult as the researcher cannot get the model structure and the parameter of the network.

Chen et al. [17] proposed their attack named Devil's Whisper which used a known model plus a modern ASR to approximate the target black-box ASR model, and found that the adversarial samples for their self-built model have good transferability on the target model.

2.2.2 Hardware-level Voice Attack

There are other attacks using ultrasound to perform an inaudible voice attack, which attacks the signal processing hardware of a voice interface. Zhang et al. [42] took advantage of the non-linearity frequency response characteristic of the amplifier in the ASR and embedded malicious commands in an inaudible ultrasonic wave. The audio command carried in the ultrasound can be recovered at the amplifier of the ASR. Yan et al. [40] proposed an attack on the devices by transmitting malicious commands in ultrasonic waves through the solid medium. Sugawara et al. [35] proposed an attack that used a laser light to control the voice control system, which is demonstrated successful attacking various commercial products at up to 110 meters with the line of sight to the target device. While hardware-level voice attacks have much better concealment than machine learning-level voice attacks, the hardware-level attacks need specifically designed devices to perform the attack, which limits the application scenario.

Chapter 3

Method

3.1 Threat Model and Assumptions

No assumption was made on the model design of the target ASR. The attack is based on the universal characteristic which exists in all of the ASR and even human comprehension, that the models pay more attention to some parts of the audio while somewhat ignoring others.

The attack was designed by inquiry. The attacker must have the ability to probe the target model as many times as they want. Each trial of attack was conducted on one command against one specific ASR system. Through one successful attack, the attacker generates an audio command which is recognized as the correct transcription by the ASR, but hard to be understood by a human.

The attack can be divided into two phases. The first phase is the preparation phase in which the attacker generates the attack audio samples by querying the target ASR system as many times as they want. The attacker does not know the model structure and can only use the target ASR as an oracle. The second phase is the attack phase in which the attacker plays the attack audio through the speaker in the victim's room. The target device is placed in the same room as the speaker. The attack is successful if the target device

correctly recognizes the audio command.

3.2 Transmission Model

Two transmission models were used in this research, over-the-line (Section 4.2, 4.3) and over-the-air (Section 4.4).

Over-the-line. The over-the-line model is shown in Fig. 3.1 a). In the over-the-line setting, the adversarial audio is directly input to the target model as a file or stream data. There is no distortion from transmission in the over-the-line setting. The over-the-line attack was used in the feasibility test.

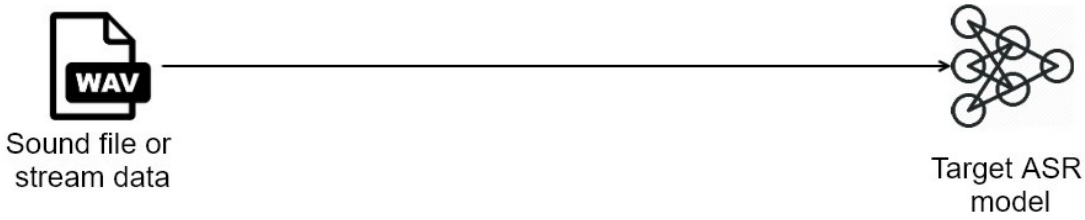
Over-the-air. The over-the-air model is shown in Fig. 3.1 b). Over-the-air is the most common and practical setting for voice attacks. In an over-the-air setting, the attacker plays the adversarial audio by an electronic speaker, and the target device records this sound with an electronic speaker and input the recording to the target ASR model. The target model could be any model capable of speech recognition, such as cloud speech-to-text API or open-source speech-to-text model. It is also possible that the microphone and the target ASR model are in an integral device, such as Google Home, Amazon Alexa, etc.

3.3 Attack Scenario

It is assumed that the target ASR is located in a room, and in this room, there is a speaker that is controlled by the attacker. The attacker is capable of playing a noise-like audio command which could be recognized as a meaningful command by the ASR, but will not be recognized as containing meaningful words by humans in the room, even if they were nearby.

The ASR system is regarded as a classifier $f_m(\cdot)$, while the input of the classifier is the waveform x of the sound signal recorded from the microphone, and the output of the

a) Over-the-line



b) Over-the-air

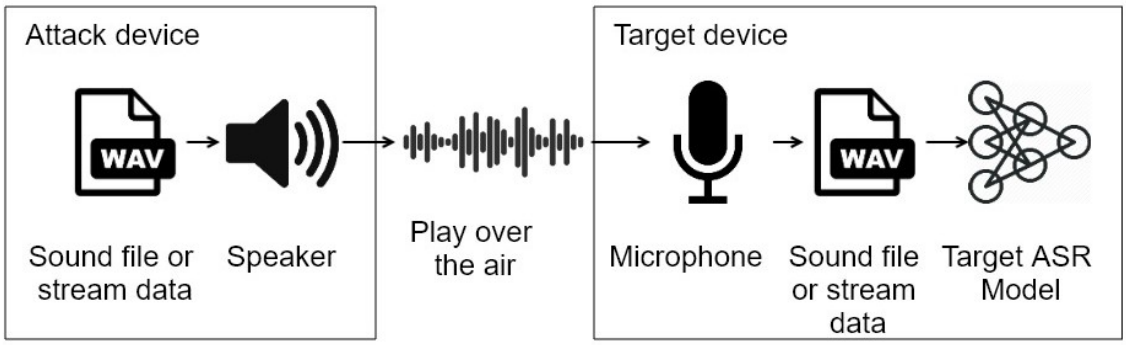


Figure 3.1: Transmission model.

ASR is a string of the transcribed result y_m. Similarly, human comprehension can also be regarded as a similar classifier $f_h(\cdot)$. The output y_h is the transcription of the waveform x from humans. Formally, this is:

$$\begin{cases} f_m(x) = y_m \\ f_h(x) = y_h \end{cases} \tag{3.1}$$

Define $S(x_0)$ as the set of the correct transcriptions of the original waveform x_0 for all of the target ASR. The correct transcription set of the waveform x_0 was acquired by inputting x_0 to all of the target ASR $f_i(\cdot)$ and adding all transcriptions into set $S(x_0)$. Use Q to denote the index number list of all of the target ASR. The correct transcription set of x_0 is formally defined as:

$$S(x_0) = \{y | y = f_i(x_0), i \in Q\} \tag{3.2}$$

In an ordinary situation, the audio can be recognized by both humans and machines. Formally, this is:

$$\begin{cases} f_m(x_0) \in S(x_0) \\ f_h(x_0) \in S(x_0) \end{cases} \tag{3.3}$$

In a hidden voice attack, the attacker crafts a bogus input x^*, which can be recognized as the correct transcription by the ASR. But a human cannot recognize the meaning of the waveform. Formally, this can be represented as follows:

$$\begin{cases} f_m(x^*) \in S(x_0) \\ f_h(x^*) \notin S(x_0) \end{cases} \tag{3.4}$$

3.4 The Perturbation Framework

Basically, the attacker tries to reduce as much information as possible from the original waveform x_0 to get a perturbed waveform x^* which can still be recognized as the correct

transcription by the machines ($f_m(x^*) \in S(x_0)$). Starting from $x^* = x_0$, we gradually increase the amount of perturbation on x^* to shift it to the classification margin f_m. If x^* has reached the classification margin, adding the smallest perturbation on x^* will make the machine generate an incorrect transcription $f_m(x^*) \notin S(x_0)$. Then we will take the x^* which is the closest to the margin and yield $f_m(x^*) \in S(x_0)$.

A schematic diagram is shown in Fig. 3.2. A small amount of perturbation was added into x_0 repeatedly, from x_1^* to x_2^* to x_n^*, until $f(x_{n+1}^*) \notin S(x_0)$. Then x_n^* will be accepted because $f(x_n^*) \in S(x_0)$ and it is the perturbed waveform that contains the least amount of information from the original waveform x_0.

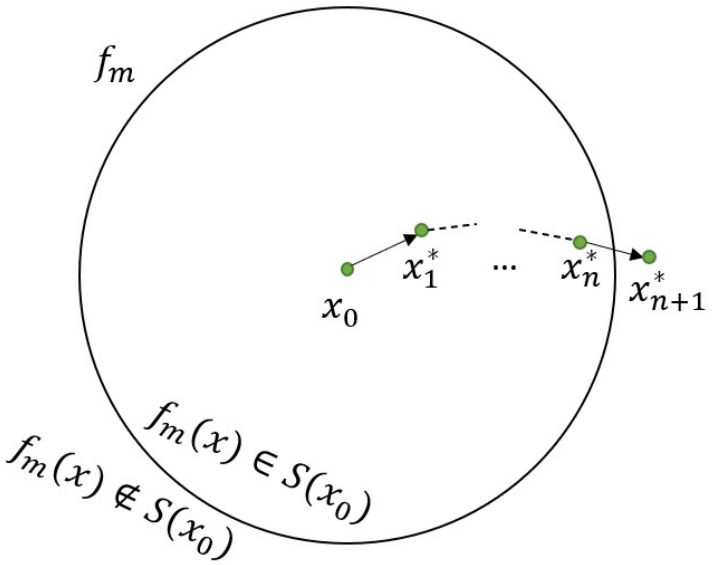


Figure 3.2: Depiction of the iterative perturbation framework.

This method can reduce the intelligibility of the sound signal while being recognized as a meaningful command by machines but it cannot guarantee that the sound cannot be understood by humans, as the human brain is capable of focusing on one of many simultaneous auditory sources. In Fig. 3.3 the left circle is the classification margin of an ASR system f_m and the right circle is the classification margin of human intelligibility f_h. The area delimited by each circle represents the space in which $f_m(x) \in S(x_0)$ or $f_h(x) \in S(x_0)$, respectively. We want to add as much perturbation on x_0 as possible to get

a x^* which approaches the classification margin but is still within its circle. The perturbed waveform can fall to x_A^* (can be recognized by the machine but also can be recognized by a human) or ideally fall to x_B^* (can be recognized by the machine and cannot be recognized by a human).

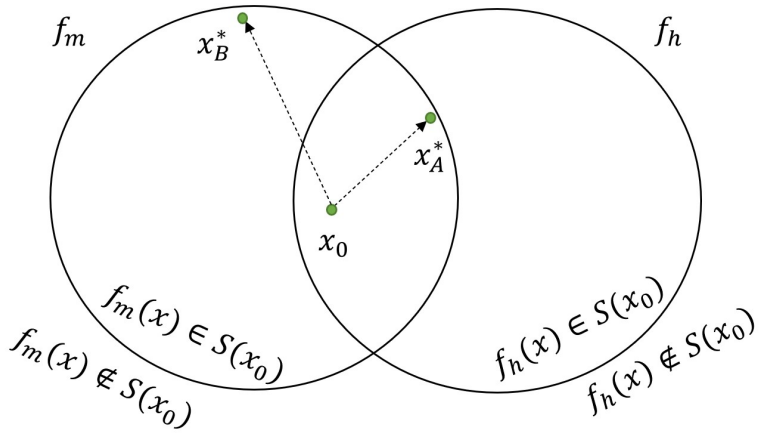


Figure 3.3: The perturbed waveform can fall to x_A^* (can be recognized by the machine but also can be recognized by a human) or ideally fall to x_B^* (can be recognized by the machine and cannot be recognized by a human).

3.5 Monotonically Increasing Perturbation Function

A Monotonically Increasing Perturbation Function (MIPF) is the core function of the perturbation framework. This function is used to add a certain degree of perturbation on the waveform x and output the perturbed waveform x^*. There should be one perturbation parameter (or a list of parameters) to control the perturbation degree of the waveform, and the relation between this parameter and the perturbation degree of the perturbed waveform should be monotonically increasing. In other words, if this parameter is slightly increased, the perturbation degree of the audio should slightly increase simultaneously.

The input of the perturbation function is the original waveform x and a perturbation

parameter p. The output of the perturbation function is the perturbed waveform x^*.

MIPF should meet the following important properties.

- If the perturbation parameter p is set to the maximum value, the waveform x^* becomes an empty signal containing no information from the original signal x.
- If the perturbation parameter p is set to the minimum value, the waveform x^* should be the same as the original waveform x.
- The relation between the perturbation parameter p and the perturbation degree of waveform x^* should be monotonically increasing. That is, if p was increased, the perturbation degree of the perturbed waveform x^* will also increase.

Technically, a very simple function called "Random Delete" (Algorithm 1) was designed as the main MIPF for our experiment. What this function does is choosing a portion of points in the waveform x and set them to 0. The input of the "Random Delete" is the waveform $x \in [0, 65535]^n$ (n is the number of points in waveform x and each point is in the format of "int16") and a perturbation rate $p \in [0, 1]$. This function randomly chooses $\lfloor np \rfloor$ non-repetitive points in the waveform x and set them to 0 to obtain the perturbed waveform x^*.

Algorithm 1 MIPF: Random Delete

Input: x, p
Output: x^*
1: $x^* \leftarrow x$
2: $n \leftarrow len(x)$
3: $n\_delete \leftarrow \lfloor np \rfloor$
4: $delete\_index \leftarrow random.sample(range(0, n), n\_delete)$
use a random sample function to get the index of the chosen points to be deleted in x.
5: $x^*[delete\_index] \leftarrow 0$
6: **return** x^*

This simple function (Algorithm 1) is an instance of MIPF. It meets the three important properties mentioned above. First, if the perturbation rate p is set to 1.0 then every point

in the perturbed waveform x^* is set to 0, thus no information of the original waveform x remains in x^*. Second, if the perturbation rate p is set to 0%, then the waveform x and the perturbed waveform x^* are the same because no point in x is deleted. Third, with the increase of p, more points in waveform x are deleted to get x^*, therefore lessening the information from x in x^*.

The "random delete" in the time domain may not be the best perturbation function for this task, but this could serve as a good baseline for evaluating the SegmentPerturb framework. Thus, a second MIPF, "random delete in spectrum" (Algorithm 2), was proposed to illustrate the practicability of SegmentPerturb as a framework. This function randomly delete the frequency components in the spectrum of the DFT. The waveform was first transformed to the frequency domain, then a proportion of the components at some frequencies was deleted (step 4 to 7 in Algorithm 2), then it was transformed back to the time domain.

Algorithm 2 MIPF: Random Delete in Spectrum

Input: x, p
Output: x^*
1: $x^* \leftarrow x$
2: $x_f^* \leftarrow \text{FFT}(x_t^*)$
3: $n = \lfloor len(x_f^*)/2 \rfloor$
4: $index\_to\_delete\_lower\_part = \text{random.sample}(\text{range}(0, n), \lfloor np \rfloor)$
5: $index\_to\_delete\_upper\_part = 2n \times 1_{1\times n} - index\_to\_delete\_lower\_part$
6: $x_f^*[index\_to\_delete\_lower\_part] \leftarrow 0$
7: $x_f^*[index\_to\_delete\_upper\_part] \leftarrow 0$
8: $x^* = \text{abs}(\text{iFFT}(x_f^*))$
9: **return** x^*

3.6 The Naïve Perturbation Algorithm

The general idea of the naïve perturbation algorithm is that a small amount of perturbation δ was added into the whole audio waveform x^* at each loop using the MIPF. At each loop, the perturbed waveform x^* was transcribed to get the result $y^* = f_m(x^*)$. If $y^* \in S(x_0)$ then this small amount of perturbation does not make the x^* fall outside the classification margin,

so this perturbation was accepted. This step was executed repeatedly until $y^* \notin S(x_0)$. The last δ would not be submitted to x^* and finally the x^* which approximated the classification margin will be outputted.

Using this basic idea and "random delete" as the core function to remove the information from x, the naïve perturbation algorithm (Algorithm 3) was proposed. The *perturbation_rate_list* in Algorithm 3 is a list of the possible values within the domain of definition of p, ordered from smallest to the largest. To find the marginal perturbation rate by traversing through the *perturbation_rate_list* from the smallest to the largest is not efficient. Therefore, in our experiment, dichotomy was applied to efficiently get the highest perturbation rate for one given waveform x_0.

Algorithm 3 Naïve Perturbation Algorithm

Input: x, y_c, f
Output: x^*

for $p \in$ *perturbation_rate_list* **do**
  $x_t^* \leftarrow \text{MIPF}(x, p)$
  $y \leftarrow f(x_t^*)$
  if $y \in S(x)$ **then**
    $x^* \leftarrow x_t^*$
  else
    break
  end if
end for
return x^*

3.7 Segmented Perturbation Algorithm

To improve the perturbation degree, a segmented perturbation algorithm (SegmentPerturb) was proposed. It is an upgraded version of the naïve perturbation algorithm. It performs more stable than the naïve perturbation algorithm, since according to the experiment in the next section, most commands can get a similar perturbation degree. Nonetheless, this new algorithm uses more inquires to the model.

The rationale for SegmentPerturb is that if an audio command was separated into

multiple segments, it is natural that the perturbation margin differs for every segment in the audio, so more perturbation can be applied to some segments which are not so important from the view of the ASR system. For example, a modern ASR based on an end-to-end deep neural network model may take each time window of the waveform as input and output the probability for each character at every time window (and then use other methods to get the final transcription, such as using a decoder and/or a language model). The command "Turn on the light", expressed in phonetic symbols is [ˈtɜːn ˈɒn ˈthə ˈlɑit], where each syllable lasts for multiple time windows. A simplified output for the neural network for each time window could look like "—ttt–uuuuuu-rr-nnnn—oo-nn-ttttt-hh-e–llll-ii-gg–hhhhh-tttt—" ("-" is the empty label). If a large amount of perturbation was added into the time window corresponding to the second "t", it is very likely that the ASR will give the same transcription. The influence on the final transcription from each time window is not the same, therefore an attacker can take advantage of this characteristic of modern ASRs and put different perturbation degrees on different segments in the audio.

Fig. 3.4 shows a schematic diagram for the perturbation boundary for every segment in a waveform. The continuous line in the diagram is the theoretical perturbation boundary which is the maximum perturbation that can be added to each segment. If the attacker applies a perturbation rate that is higher than the red line boundary at any segment, the ASR system will generate an undesirable transcription ($f(x) \notin S(x_0)$). But if the attacker applies a perturbation rate that is slightly lower than the theoretical perturbation boundary for every segment (the blue bar in Fig. 3.4), the highest average perturbation rate (APR) for the waveform can be approximated.

According to this theory, SegmentPerturb (Algorithm 4) was designed to get the highest perturbation rate for every segment. First, the audio waveform was separated into multiple equal-length segments. Then starting from the smallest perturbation rate, this perturbation rate was applied to one segment in x_0, and then it was transcribed using the ASR. If the ASR system outputted a correct transcription ($f_m(x^*) \in S(x_0)$), this small perturbation on this segment would be accepted; otherwise, this change would not be accepted (and a flag

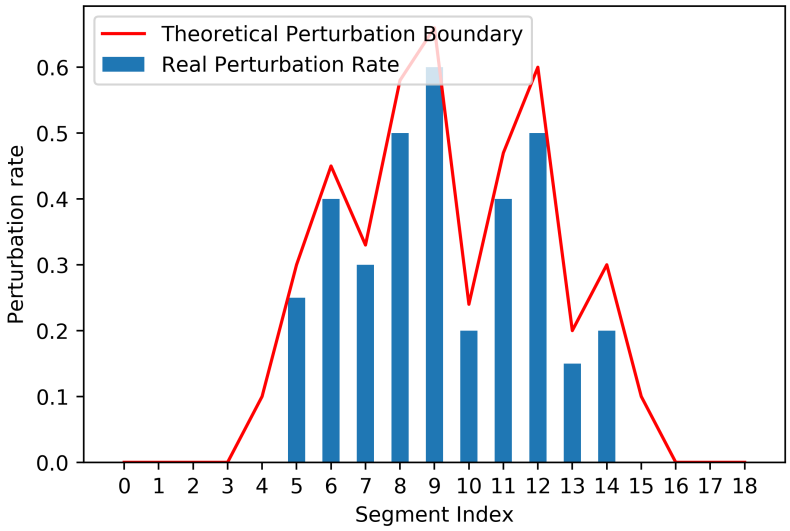


Figure 3.4: The schematic diagram for SegmentPerturb.

"*meet_end*" would be used to mark that the perturbation rate for this segment has reached the perturbation boundary). The same procedures were repeated on all segments. Then the perturbation rate gradually increased until all of the segments have met the perturbation boundary or all of the perturbation rates have been traversed.

There are many segments in the waveform x_0 which contain only 0. If the perturbation were applied at these segments, the outcome will be meaningless. Therefore, these segments were excluded using a simple threshold, to get the segments with the voice in them. Using a simple threshold is efficient since the original command waveform x_0 contains no ambient noise. To get a better result, the *segment_list* has a random sequence.

Algorithm 4 SegmentPerturb

Input: x, y_c, f
Output: x^*

$x^* \leftarrow x$
perturb_rate_list is a list containing a sequence of ordered perturbation rates, for example [0.1, 0.2, 0.3, ... 0.9, 1.0].
segment_list is the segments in x which contain voice (random sequence).
meet_end is the vector marking whether the perturbation rate of every segment has reached the perturbation boundary. It is initialized as all $False$.
for $p \in perturb\_rate\_list$ **do**
  for $b \in segment\_list$ **do**
    if $meet\_end[b] == False$ **then**
      $x_t^* \leftarrow x^*$
      $x_b^* \leftarrow x_b$
      # Take out the segment b for perturbation.
      $x_b^* \leftarrow \text{MIPF}(x_b^*, p)$
      # Only apply the MIPF on the segment b
      replace the segment b in x_t^* with the x_b^*
      $y \leftarrow f(x_t^*)$
      if $y \in S(x)$ **then**
        $x^* \leftarrow x_t^*$
        # Accept this perturbation rate and submit the change.
      else
        $meet\_end[b] \leftarrow True$
      end if
    end if
  end for
end for
return x^*

Chapter 4

Experiment

4.1 Experiment Setup

Hardware. The devices used in the experiment are ordinary devices that may be used domestically. A personal laptop was used for the experiment. The microphone in the experiment is an ARCHEER condenser microphone with a frequency response of 20 Hz to 20 kHz, a sensitivity of -34 dB $\pm$ 2 dB, and a signal to noise ratio of 78 dB. The electronic speaker in the experiment is EDIFIER R980T, which is an ordinary speaker with a signal-to-noise ratio of $\geq$ 85 dBA, distortion of $\leq$ 0.5%, and frequency response of 70 Hz to 20 kHz.

Target Model. The target models were four commercial speech-to-text APIs plus one smart home device. The four commercial speech-to-text APIs were Google Speech Recognition [2], Microsoft Azure speech-to-text API [5], Wit API [7], and IBM speech services [3]. All of the speech recognition APIs mentioned above can be used through online requests. The smart home device used in the experiment was Google Home. The experiment is conducted on the most popular speech recognition APIs to verify the practicability of the attack in the real world.

Command selection and audio acquire. A representative command set of six commands are used in the experiment. The commands are "Turn on airplane mode", "Open the door", "Turn on the computer", "Turn on the light", "Call 911", "Turn on Wi-Fi". A similar command set was used in previous works [8, 13, 17] to represent a variety of the commands on the ASR system. The voice commands were recorded in a quiet room.

Criteria for the correctness of the ASR and human transcription. For ASR, a "correct transcription" for the perturbed audio file should be exactly the same as the transcription of the original audio file by the same ASR. For example, if the original audio file of "Turn on airplane mode" was input in Azure speech recognition API, the API will return a string of "Turn on airplane mode." (the first character is uppercase and there is a period in the end). The transcription for an attack sound sample is regarded as correct only if the API returns exactly the same string as the transcription of the original command. Other strings such as "Turn on the airplane mode" (an extra "the") are regarded as an incorrect transcription even if it had the same meaning.

For the human transcription of the audio, the criteria were less rigorous. The "correct transcription" for human transcription should be a verbatim match with the command but ignore cases and punctuation. For example, for the command "Open the door", the participants in the audio intelligibility test may input "OPEN THE DOOR", "open the door" or "Open the door." which were all regarded as correct transcriptions.

4.2 Preliminary Experiment: Over-the-line Attack using Naïve Perturbation Algorithm

The naïve perturbation under the over-the-line setting was tested as a preliminary experiment. This experiment was conducted on every command against every speech-to-text API. The perturbation rate for every command list against all of the ASR is shown in Fig 4.1.

For Google Speech API, most of the commands got a relatively high perturbation rate, apart from IBM, the perturbation rate for all other APIs got more than 0.5. "Turn on airplane mode", "Call 911" and "Turn on Wi-Fi" even get around 0.9 perturbation rate, which means that even 90% of the points in waveform x are set to 0, the Google Speech API can still recognize it as the correct transcription. The naïve perturbation could only perturb as much as 70% and 15% on the Wit and the IBM speech recognition APIs respectively.

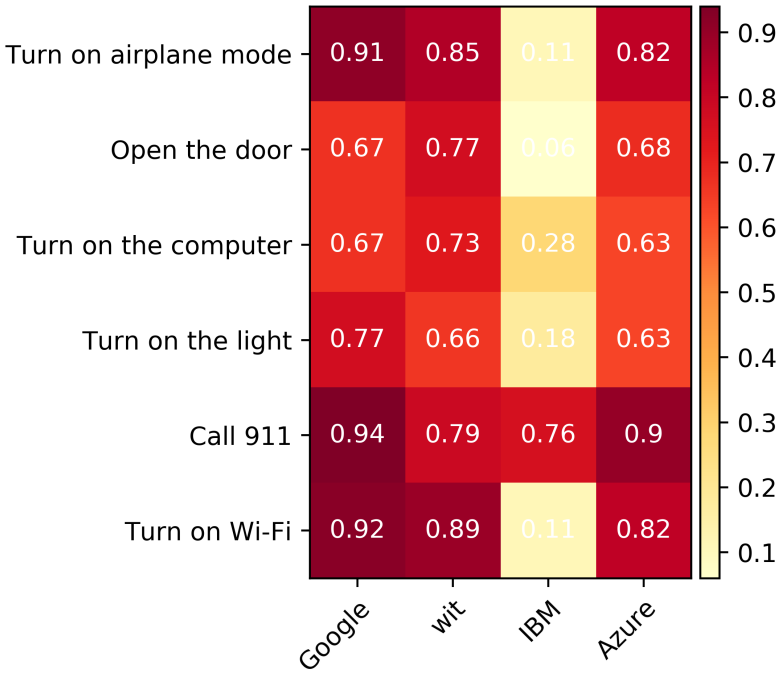


Figure 4.1: Perturbation rate for the over-the-line attack using naïve perturbation.

4.3 Over-the-line Attack using SegmentPerturb

SegmentPerturb framework was tested using different metrics to represent the efficiency of the attack. Two different MIPF, "random delete" and "random delete in spectrum", were included in the experiment.

4.3.1 Over-the-line Attack using "SegmentPerturb - Random Delete"

Heatmap The Average Perturbation Rate (APR) was calculated for every command against each speech-to-text API (Fig. 4.2). APR is the average of the perturbation rate of all the segments that participated in the perturbation procedure.

As shown in Fig. 4.2, all of the APRs for attacking Google, Wit and Azure are higher than 0.8. Those audio samples are very difficult to understand even if replayed multiple times (see Section 4.6), which indicates that this method is very successful in the over-the-line experiment.

Compared with Fig. 4.1 for naïve perturbation, the APR in Fig. 4.2 for all of the speech APIs showed steady improvement. Especially for the IBM speech services, in Fig. 4.1 the perturbation rate for most of the commands on IBM is lower than 0.3, which indicates that it is easy for humans to understand the perturbed voice command. But using SegmentPerturb, the APR exceeds 0.5 for all of the commands, some commands even above 0.8.

Segment length - APR The relationship between segment length and APR was illustrated in Fig. 4.3. The attack audio samples for every command were generated twice against the Google speech recognition API, and use the APR to draw the boxplot figure. The segment length was set from 10 ms to 50 ms.

The mean of APRs at a smaller time window is slightly higher than using a bigger window, and the variance is smaller with a smaller time window. If the audio is separated into smaller segments to perform the perturbation, more perturbation can be added to the unimportant segments. But if the segment length was too small (less than 20 ms), too many inquiries would be made to the target model. Therefore, 20-30 ms could be a good segment length for SegmentPerturb.

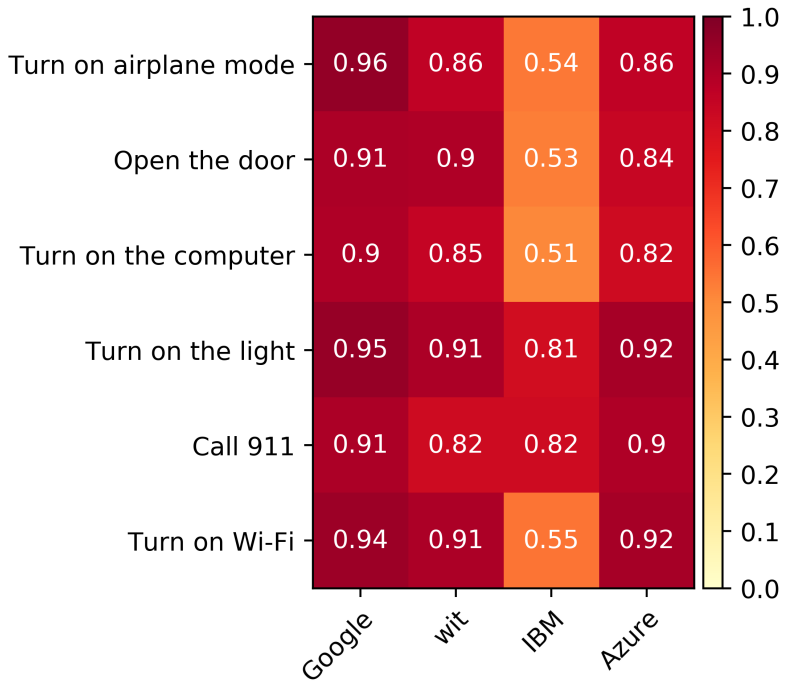


Figure 4.2: The APR for over-the-line attack using "SegmentPerturb - Random Delete".

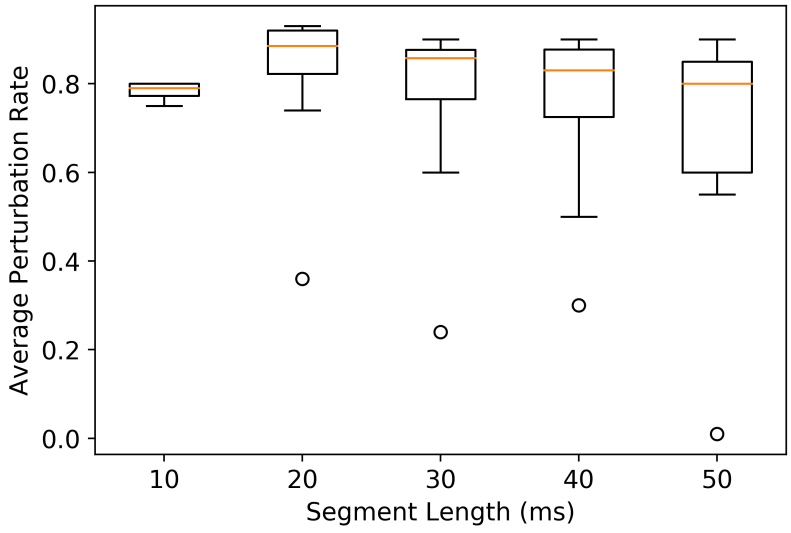


Figure 4.3: The relationship between segment length and APR for "SegmentPerturb - Random Delete".

4.3.2 Over-the-line Attack using "SegmentPerturb - Random Delete in Spectrum"

Following the same SegmentPerturb framework, the "Random Delete in Spectrum" was applied as the MIPF, to test the versatility of the SegmentPerturb framework.

Heatmap The APR for the SegmentPerturb using "Random Delete in Spectrum" as MIPF was shown in Fig. 4.4. The audio for each command is separated into 30 ms time windows. SegmentPerturb was applied to all commands against all ASRs. All commands got an APR of around 0.8, which suggests that using "random delete in spectrum" can be a more stable MIPF for SegmentPerturb's framework.

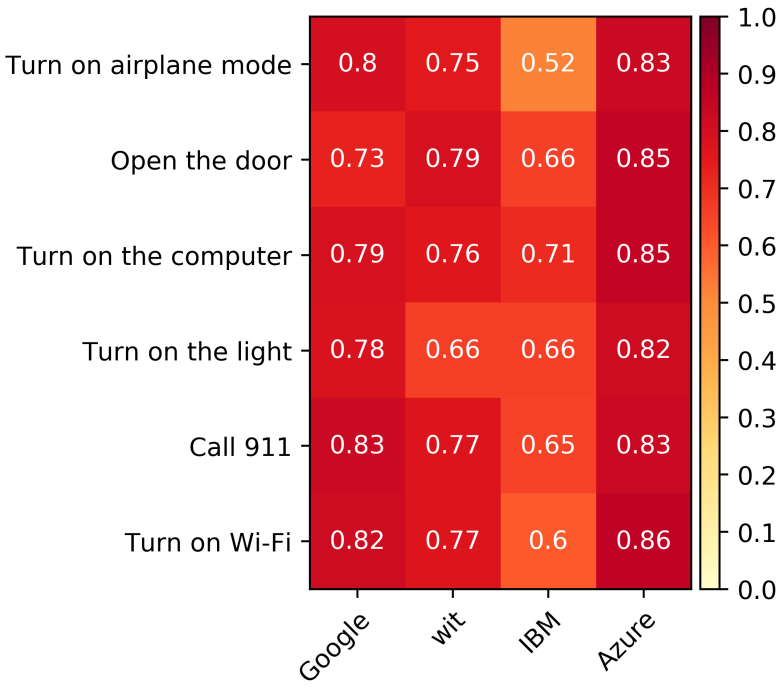


Figure 4.4: The APR for over-the-line attack using "SegmentPerturb - Random Delete in Spectrum".

Segment Length - APR The relationship between the Segment Length and APR for the over-the-line attack was studied. The segment length range from 10 ms to 50 ms.

"SegmentPerturb - Random Delete in Spectrum" was applied to generate the attack samples for all of the commands twice against the Google API and the APR was recorded. The result is shown in Fig. 4.5. The segment length and the APR have a negative correlation, which means that if the segment is smaller, more perturbation can be added into the audio samples. The APR between 20 and 40 ms is similar, with an average of 0.81. A theoretical explanation for the result at 20 to 40 ms could be that the modern ASRs normally use a time window of around 20 ms to 40 ms. If a similar time window is applied to generate the hidden voice command, the hidden voice command can be easier recognized by the target model. If the segment length were too small there would be too many inquiries to craft one attack audio sample. Therefore 20 - 30 ms should be a reasonable time window for SegmentPerturb.

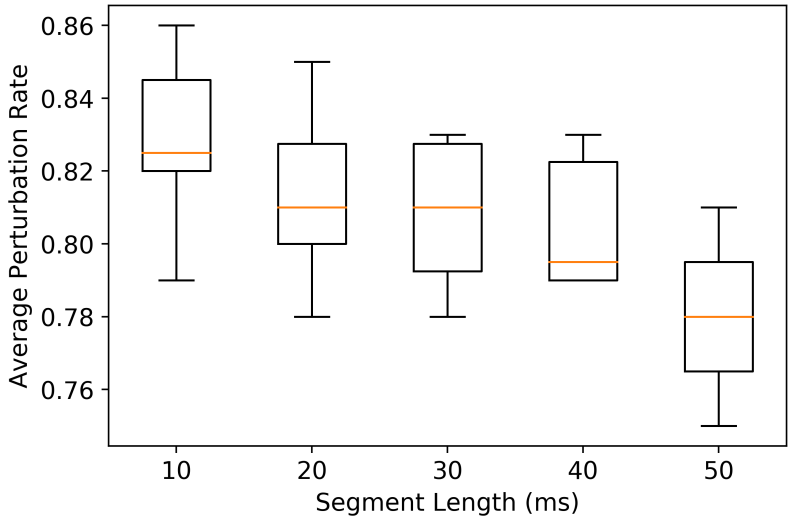


Figure 4.5: The relationship between segment length and APR for "SegmentPerturb - Random Delete in Spectrum".

4.4 Over-the-air Attack using SegmentPerturb

4.4.1 Over-the-air Attack using "SegmentPerturb - Random Delete"

Heatmap SegmentPerturb was in the over-the-air setting in this section. The parameters were the same as in Section 4.3; a 20 ms time window was used and the *perturb_rate_list* was [0.2, 0.4, 0.6, 0.8, 0.9, 1.0]. The distance between the speaker and the microphone is one meter.

Fig. 4.6 shows the APR of the over-the-air attack experiment. For over-the-air attacks, the APR ranges from 0.50 to 0.92. IBM speech API can hardly recognize any original audio from over the air. Therefore, the perturbation rate tolerated by IBM was low. The amount of perturbation added into the audio command was highly related to the abnormal tolerance of the ASR. For each command, more perturbation can be added into the audio transcribed by Google, Wit, and Azure speech-to-text API.

Perturbation rate for every segment. The perturbation rate for every segment of the waveform was analyzed in this section. The first audio command "Turn on airplane mode" was randomly chosen to conduct this experiment. Results are shown in Fig. 4.7.

The waveform of the command was replayed in slow motion and it was found that from 1.1s to 1.6s corresponds to "turn on", from 1.7s to 2.0s is for "airplane", and from 2.0s to 2.5s is for "mode". All of the segments which are the valid parts of the waveform took part in the perturbation procedure (from 1.1s to 2.5s). The segments whose amplitude is lower than the threshold did not participate in the perturbation procedure; these are shown in light yellow (the perturbation rate for these segments is 0). Some parts which should not be activated as the valid segments were activated because of the noise in the room (from 0s to 1s). But these segments only take a small proportion of all of the segments and will not make a difference in APR. In Fig. 4.7 Google, Wit and Azure got high perturbation rates

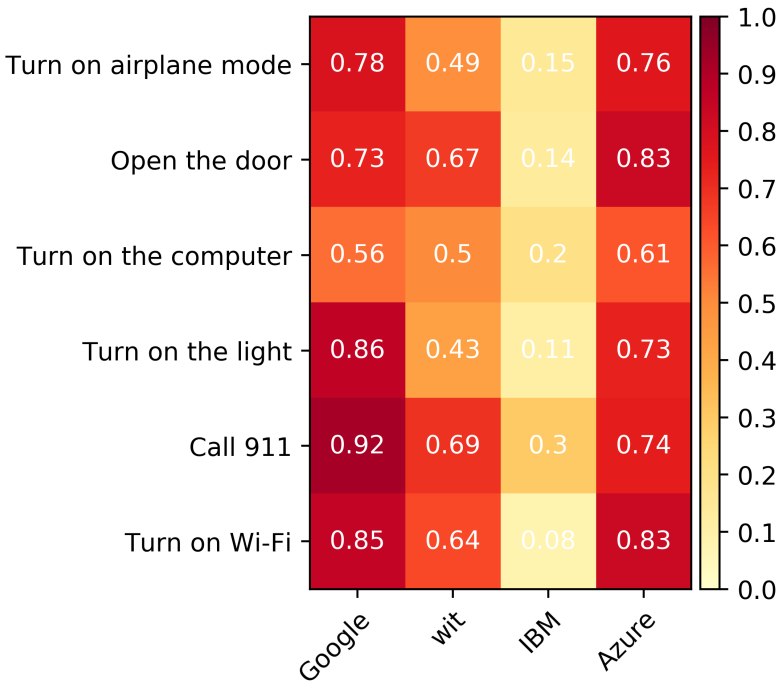


Figure 4.6: The APR for over-the-air attack using SegmentPerturb.

throughout the signal. The figure showed that the segments which are deemed important from the view of the machine are not the segments that are important from the view of the human. Humans pay more attention to the stress of the word (peak of the signal), but those parts can be added more perturbation with SegmentPerturb.

Attack distance - APR. The performance of the attack in different ranges of distances was tested in this section. The Google speech recognition API was used in this test. The distance from the speaker to the microphone was the variable of the experiment and the APR for generating the audio command samples was observed. The result for the distance and APR is illustrated in Fig. 4.8. When the distance is one meter, the mean APR is around 0.8. From 1.5 m to 2.0 m the mean APR remains above 0.7. As the distance increases to 2.5 meters, the mean APR remains at 0.6. According to the audio intelligibility in the next section, if the APR is higher than 0.7, the probability for a human to comprehend the meaning of the command could be very low. Therefore, two meters seems to be an effective

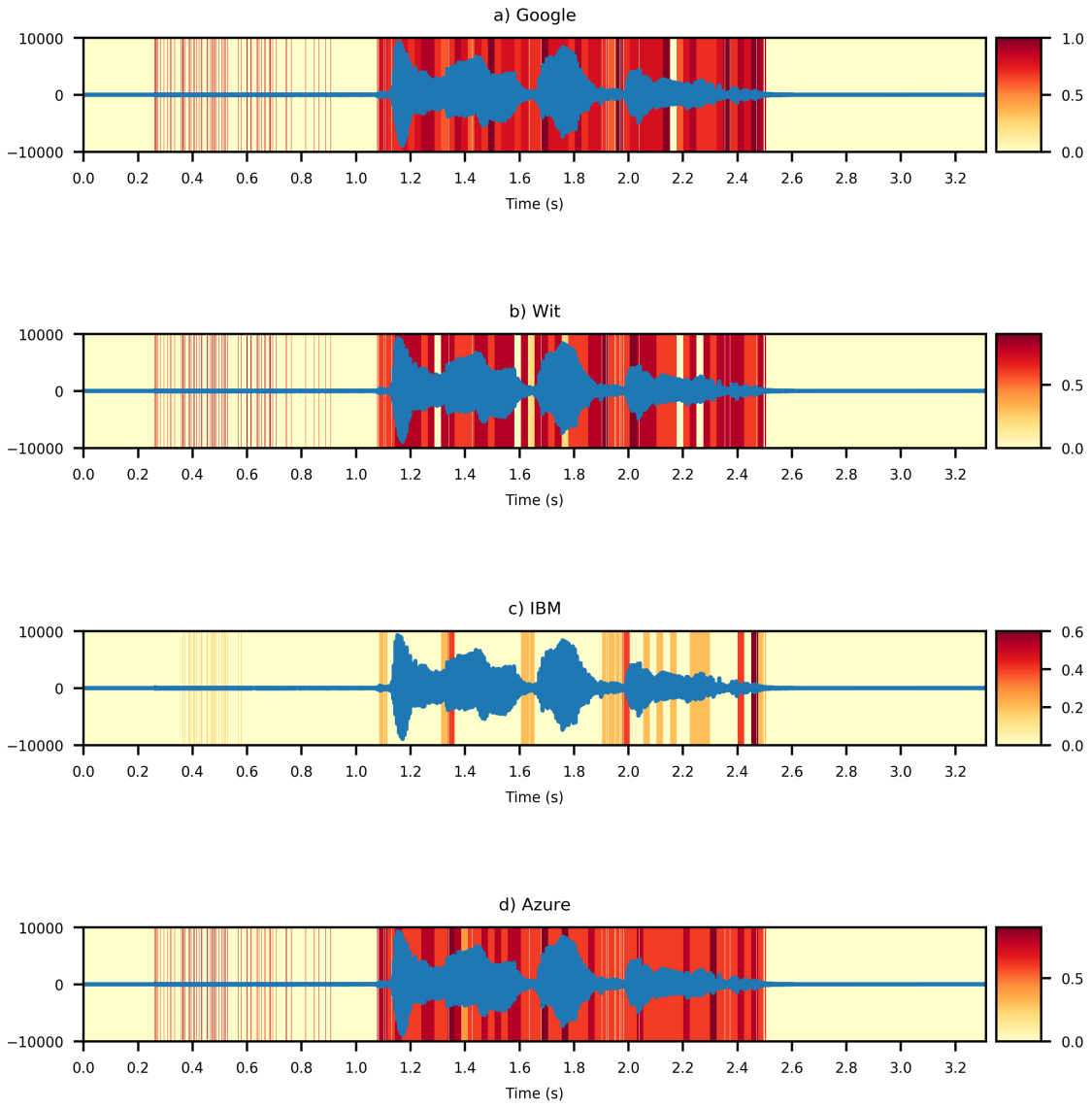


Figure 4.7: The perturbation rate for every segment of the command "Turn on airplane mode".

distance for our attack.

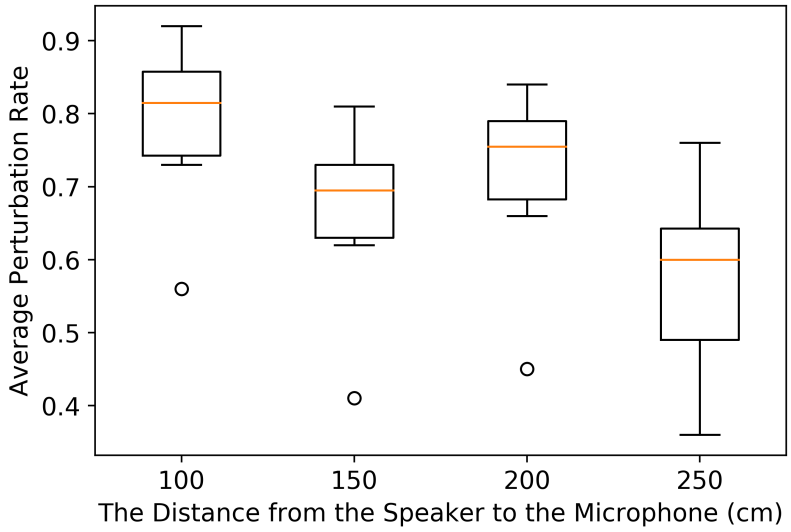


Figure 4.8: The relationship between the physical distance and the APR.

Attack on Speech Recognition API The audio samples crafted from the over-the-line were applied to attack the target model from over-the-air. The distance from the speaker to the microphone was set from 50 cm to 200 cm. Each command was played five times, and the ratio of trials correctly recognized by the speech recognition API was recorded and shown in Table 4.1. The acceptance rate for the audio command samples generated using "SegmentPerturb - Random Delete" is significantly dropped from around 35% at 50 cm to 17% at 200 cm. The audio command samples have some intelligibility from the machine.

Distance (cm)	50	100	150	200
Accept Rate	33%	37%	17%	17%

Table 4.1: Over-the-air attack on Google speech recognition API, using "Segment-Perturb - Random Delete"

Transferability Test The result of the transferability test is demonstrated in Table 4.2. The attack audio samples from Section 4.4 were used to perform a replay attack over-the-air

against all the other target speech recognition APIs. Each adversarial sample was played five times and the ratio of the audio samples which can be successfully recognized by the other speech-to-text APIs was recorded and presented at the corresponding position in Table 4.2. The row for IBM was not concerned because the APR of attacking IBM was too low that it should be easily recognized by a human.

SegmentPerturb is originally designed to explore the vulnerability of one specific target ASR system, so the anticipated transferability of the SegmentPertrub would not be very strong. Besides, even if the experiment took place in a quiet room the ambient noise cannot be removed completely. The result shows that SegmentPerturb has some transferability. If the APR is lower for the attack on one ASR, those adversarial samples are a little clearer and can be recognized by other ASRs which has higher abnormal tolerance. For example, the adversarial audio on Wit has an average APR of 0.57, 87% of the attack can be transferred from Wit to Azure. Google speech recognition API and Azure speech-to-text API both have high abnormal tolerance, therefore 20% of the adversarial samples can be transferred from Google to Azure, and 56% can be transferred from Azure to Google. Both Google and Azure can recognize a very messy command which is used to attack other API.

		To			
		Google	Wit	IBM	Azure
From	Google		0%	0%	20%
	Wit	30%		0%	87%
	IBM	-	-		-
	Azure	56%	3%	0%	

Table 4.2: Transferability test for the attack.

4.4.2 Over-the-air Attack using "SegmentPerturb - Random Delete in Spectrum"

In this section, the MIPF of the SegmentPerturb framework was replaced with "Random Delete in Spectrum".

Attack on Speech Recognition API An over-the-air attack on the Google speech recognition API was performed. Each command was played ten times over the air, and the number of times that the audio was correctly transcribed by the speech recognition was recorded. The number of correct transcriptions was divided by the number of trials to get the acceptance rate of the attack. The result is shown in Table 4.3. From 50 to 100 cm the attack has an acceptance rate of around 65%, and at a distance of around 150 to 200 cm the acceptance rate decreased to around 39%, which is still significant, considering that this is a black-box attack.

Distance (cm)	50	100	150	200
Acceptance rate	67%	63%	33%	43%

Table 4.3: Over-the-air attack on Google speech recognition API, using "Segment-Perturb - Random Delete in Spectrum".

4.5 Practicability Test on Google Home

4.5.1 Attack Distance

The feasibility of the attack was tested with Google Home. Since it is not feasible to get the transcription from the Google Home device, smart plugs with different names ("airplane mode", "computer", "light", "Wi-Fi") was set up to test if the hidden voice command was correctly transcribed. If the command was correctly recognized by Google Home, the smart plugs would be switched on.

The attack samples[1] obtained from attacking Google speech-to-text API over-the-air using "Random Delete" was used in attacking the Google Home. Only the "turn on [device]" commands were tested, because if the smart plug was turned on, it is assured that the command is correctly executed by Google Home. For other commands, Google Home will

[1]The samples can be found in this website `https://walterjohnson0.github.io/AudioClips.html`, we recommend the reader to try out our hidden voice command using Google's smart home device plus smart plugs.

reply "Sorry, I cannot understand" for "Open the door". Besides, we do not want to make accidental calls to the 911 emergency services for testing the "Call 911". Therefore, "Open the door" and "Call 911" were excluded from the command set.

The first experiment was on the attack distance and the volume of the speaker. The attack audio sample can be recognized by Google Home from a variety of distances. But, naturally, if Google Home is placed far away from the device, the volume should be increased. Therefore, the minimum system volume for Google Home to recognize the command was recorded. The result is shown in Table 4.4. The attack audio samples can be recognized by Google Home from up to 2 meters, but the volume of the speaker should be loud enough.

Command (APR)	Minimum System Volume			
	50 cm	100 cm	150 cm	200 cm
Turn on airplane mode (0.78)	20	40	50	50
Turn on the computer (0.56)	20	30	40	40
Turn on the light (0.86)	30	50	50	50
Turn on Wi-Fi (0.85)	20	20	40	40

Table 4.4: Attack on Google Home. The minimum system volume for the hidden voice command to be recognized.

4.5.2 Real Scenario Test

To test the feasibility, a real scenario attack was performed in an actual bedroom. The bedroom's floor plan is depicted in Fig. 4.9. Google Home was placed at positions A to F. The speaker was placed at position S. The distance from the speaker S to each position was S-A (50 cm), S-B (220 cm), S-C (150 cm), S-D (270 cm), S-E (270 cm), and S-F (370 cm). All positions were at the same horizontal plane which is 75 cm above the ground, common height for an office desk. The system volume was set at 60. The result regarding whether the attack audio was recognized by Google Home is shown in Table 4.5. Most commands can be recognized by Google Home at each position. At position A, the command was too loud that one of the commands was not recognized by Google Home. To our surprise, at position F, the farthest position, all commands were correctly recognized by Google Home.

Therefore, a real-world attack using SegmentPerturb could be feasible.

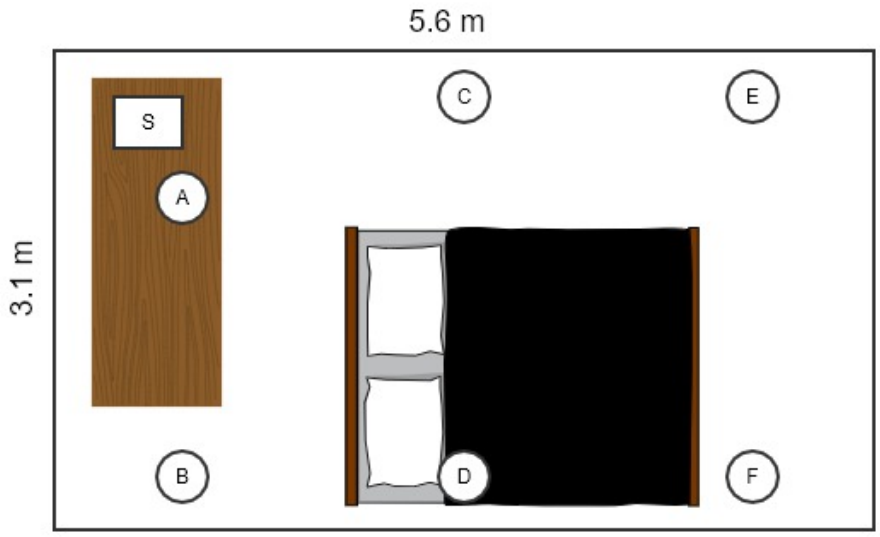


Figure 4.9: A floor plan of the bedroom. Google Home was placed at positions A through F.

Command (APR)	Position					
	A	B	C	D	E	F
Turn on airplane mode (0.78)	✗	✓	✓	✗	✓	✓
Turn on the computer (0.56)	✓	✓	✓	✓	✓	✓
Turn on the light (0.86)	✓	✓	✓	✓	✗	✓
Turn on Wi-Fi (0.85)	✓	✗	✓	✗	✓	✓

Table 4.5: Real scenario attack on Google Home. ✓: The command can be recognized by Google Home. ✗: The command cannot be recognized by Google Home.

4.6 Audio Intelligibility Study

The intelligibility experiment[2] was conducted in the form of a questionnaire in Amazon Mechanical Turk to evaluate the intelligibility of the audio commands we designed. The goal

[2]The experiment was approved by the Research Ethics Board of our institution.

of the experiment is to test the intelligibility of the normal audio clips and the Segment-Perturb audio clips. The audio intelligibility test uses the attack audio samples[3] obtained from attacking Google speech-to-text API over-the-air in Section 4.4.

250 participants were recruited to participate in the intelligibility study. The participants are assigned three groups (pool A, pool B, and pool C), 100 participants for pool A and B, and 50 participants for pool C. Participants from pool A received the following command sequence: [Turn on airplane mode (obfuscated), Open the door (normal), Turn on the computer ("Random Delete"), Turn on the light (normal), Call 911 ("Random Delete"), Turn on wireless hotspot (normal), How are you (validation), Turn on Wi-Fi ("Random Delete")]. Participants from pool B received another command sequence: [Turn on airplane mode (normal), Open the door ("Random Delete"), Turn on the computer (normal), How are you (validation), Turn on the light ("Random Delete"), Call 911 (normal), Turn on wireless hotspot ("Random Delete"), Turn on Wi-Fi (normal)]. The participants from pool C received the following command sequence: [Turn on airplane mode ("Random Delete in Spectrum"), Call 911 ("Random Delete in Spectrum"), Good morning (validation), Turn on the light ("Random Delete in Spectrum"), Open the door ("Random Delete in Spectrum"), Turn on Wi-Fi ("Random Delete in Spectrum"), How are you ("validation"), Turn on the computer ("Random Delete in Spectrum")]. The validation "how are you" command is a normal audio clip used to verify that the participant had put the proper attention in participating in the study. In general, the participants received either the normal or the obfuscated version of one command. Besides, the participants were kept engaged because they alternately got normal or obfuscated commands in the questionnaire. The questionnaire for this experiment is in the Appendix.

Participants were asked to play each command twice and input what they were able to understand each time. The whole survey lasted for about four minutes. Participants

[3]All of the audio clips for this experiment are available in `https://walterjohnson0.github.io/AudioClips.html`. Please be aware that it may be easier for readers to comprehend the meaning of the audio clips because they have already read about the list of commands. But the participants in this study were not given the list of commands in advance.

Command (APR)	Normal		Random Delete		Perturb in Spectrum	
	1 st	2 nd	1 st	2 nd	1 st	2 nd
Turn on airplane mode (0.78)	63%	76%	1%	2%	0%	5%
Open the door (0.73)	96%	100%	27%	38%	75%	88%
Turn on the computer (0.56)	92%	96%	42%	57%	41%	54%
Turn on the light (0.86)	93%	96%	20%	32%	60%	75%
Call 911 (0.92)	93%	93%	15%	22%	68%	73%
Turn on Wi-Fi (0.85)	96%	100%	45%	65%	39%	59%

Table 4.6: Audio Intelligibility Test.

were compensated with CA$0.92 according to the minimum wage in Ontario. The result eliminated the data from those participants who did not obtain a correct answer for the validation audio or report a technical issue (there was a question at the end of the questionnaire to determine whether the participant encountered technical issues).

The number of participants who inputted the correct transcription was recorded and it was divided by the number of validated data in that pool to get the percentage of the participants who recognized the command. These percentages represent the intelligibility of the command. Results are shown in Table 4.6. All of the normal commands obtained a high value in intelligibility, which indicates that those commands are clear, while obfuscated commands got much less in intelligibility. SegmentPerturb using random delete reduced around 64% the participants' intelligibility of the obfuscated audio.

Chapter 5

Discussion

5.1 Comparing with Previous Works

Following the comparison table from Abdullah et al. [9], a comparison with other black-box voice attacks [8, 11, 13, 17, 35, 37, 42] was conducted in the following dimensions: the audio type of the adversarial audio, the transmission medium of the adversarial audio, the number of queries, the time needed to craft each adversarial command, the number of ASR successfully attacked, the model type (interval) of the ASR, the attack distance, and the transferability of the adversarial command. Table 5.1 summarizes how SegmentPerturb compares against other attacks found in the literature. In general, the number of queries of SegmentPerturb is acceptable and it achieves a further attack distance than the previous attacks, however the limit for this attack is the transferability.

Attack Method	Audio Type	Medium	# Queries	Time	# ASR Attacked	ASR internals	Attack distance	Transferability
Devil's Whisper [17]	Clean	L/A	1500	4.6h	3	?	0.05-2 m	Yes
Alzantot et al. [11]	Clean	L	?	37s	1	CNN	N/A	No
Dolphin attack [42]	Inaudiable	A	0	0	9	RNN, ?	1.5 m	Yes
Light command [35]	Inaudiable	A	0	0	4	?	110 m	Yes
Abdullah [8]	Noisy	L/A	10	Seconds	12	RNN,HMM,?	0.3 m	Yes
Cocain Noodles [37]	Noisy	L/A	?	?	1	?	0.3 m	No
Hidden voice command [13]	Noisy	L/A	?	32h	1	RNN	0.5 m	No
SegmentPerturb	Noisy	L/A	Around 300	30min	4	?	0.5-2 m	No

Table 5.1: Comparing with previous work. 1: Audio Type, the audio type of the adversarial audio. 2: Medium: "L", over-the-line; "A", over-the-air. 3: # Queries: number of queries on the ASR to generate the attack model or audio. "?" means that the author did not provide this information. 4: Time: time consumption for generating one adversarial audio clip. 5: # ASR Attacked: number of ASR which is demonstrated successfully attacked. 6: ASR internal: the internal model type of the ASR. "?" means that the model type is unknown as the attack is in a black-box setting. 7: Attack distance: the workable over-the-air attack distance from the speaker to the microphone. 8: Transferability: the transferability of the attack.

5.1.1 Types of Adversarial Audio

Different attack methods use different types of audio. The types of the audio type of adversarial audio can be categorized as inaudible, noisy, clean.

- **Inaudible:** The inaudible type means that the attack cannot be heard by the human, but it can be recognized by the ASR. The attack is not necessarily in the form of audio. All of the voice attacks using inaudible methods are hardware-level voice attacks, software level attack only generate the audio which is audible. The Dolphin attack [42] took advantage of the non-linearity of the amplitude of the ASR, and generate ultrasound as the adversarial audio which is beyond the frequency range that humans can perceive (20 Hz to 20 kHz). The light command [35] is not in the form of sound. Instead, it used the characteristic of the microphone that it may incidentally respond to the light signal and used laser light to activate the ASR, which is not in the form of sound.

- **Noisy:** The noisy type implies that the adversarial audio sounds like noise to a human, but the audio is carefully designed and can be recognized by the ASR. The category of the attack which generates this kind of audio is the "hidden voice attack". Our attack is in this category. The general approach for a hidden voice attack is that the adversary first acquires the normal command, performs some transformation to this command which interferes with human intelligibility, but maintains ASR intelligibility [8, 13, 37]. Otherwise, the adversary may start from the machine output, try to reverse the input of the machine, normally this input cannot be recognized by a human.

- **Clean:** The clean type of audio is clean to humans, such as music or dialogue, but it would be recognized as a malicious command by the ASR. The command recognized by the ASR is different from the meaning of the audio to a human. The clean audio is generated by the adversarial attack. The general approach for this attack is starting

with a normal audio (the clean audio to be heard) adding the smallest modification which can be perceived by the ASR but hard to be perceived by humans [33]. Most of these types of attacks are using a white-box model on the ASR. But a recent study by Chen et al. [17] manage to use a shadow model to approximate the target model and therefore it is workable in a black-box setting.

5.1.2 Computational Power Consumption

Number of Queries

For the black-box attack, the adversary may need the ability to queries the ASR to gain the needed information to generate the adversarial audio. Chen et al. [17] used a white-box model as a local model and query the target ASR 1500 times to complement the local model. The hardware attack (Dolphin attack [42], Light command [35]) are attacking the signal processing stage of the ASR, the attack medium (ultrasound, laser light) contains the full information of the original command but the medium itself cannot be perceived by a human. A hardware attack does not involve gaining information from the ASR, therefore the number of queries is zero. For a hidden voice attack in the black-box setting, the adversary has to query the ASR as the feedback for crafting successful adversarial samples, therefore Vaidya et al. [37], Abdullah [8] and our approach require the ability to query the ASR.

Time to Generate Adversarial Samples

To generate one adversarial sample, the adversary may need to build a shadow model or repeatedly query the target ASR. The time for generating one adversarial sample is reflecting the efficiency of the attack. Devil's Whisper [17] requires approximately 4.6 hours for tuning the model and generating an attack audio sample. The hardware does not require extra time to generate the audio. SegmentPerturb needs around half an hour to generate adversarial audio, which is better than [17] and [13].

5.1.3 ASR Attacked

To prove the feasibility of the attack, the efficiency of SegmentPerturb is demonstrated with four modern commercial ASRs. The papers which also demonstrate their attack with modern ASRs are Devil's Whisper [17], Dolphin attack [42], Abdullah et al. [8]. The internal of the ASR includes CNN, RNN, HMM, and unknown ("?" in Table 5.1; it is used to mark the attack in the black-box setting). Carlini et al.'s [13] demonstrated their work with a white-box ASR (the internal is RNN), and then generalized it to the black-box model. Other works, Devil's Whisper [17], light command [35], Abdullah et al.'s [8] and SegmentPerturb are designed for attacking the ASR in a black-box setting, which could make the attack more practical in the real world.

5.1.4 Transferability

Transferability is the ability that an adversarial sample designed for one ASR can be used to attack other ASR. SegmentPerturb is specifically designed for the attacked ASR therefore the transferability of the attack is weak. Other machine learning level attack has been demonstrated some transferability, but the transferability of the attack on the neural network of the ASR is elusive because the update of the commercial ASR could make the successful Adversarial sample of the previous version fail. The hardware attack is transferable because it does not rely on deceiving the neural network model of the ASR.

5.2 Detail Comparison with the State-of-the-art Hidden Voice Attack

The attack from Abdullah et al.'s [8] is reproduced in this section according to the code provided by the authors[1].

[1]The link provided in [8] is `https://github.com/hamzayacoob/VPSesAttacks`

5.2.1 Background

The authors provided the code of their four attacks, which are time domain inversion (TDI), random phase generation (RPG), high frequency addition (HFA), and time scalling (TS). However, they did not provide the code of their so-called "perturbation engine" they refer to in [8], which is meant to combine the four attack methods to generate the "best attack sample" or the "best parameter set" for their attack. In lieu of the perturbation engine, the evaluation was done on each attack method they proposed, which in itself represented the actual attack efficiency an attacker would experience in crafting this attack.

Before reproducing these four attacks, it was important to clarify the parameters of each of them. The first attack method they proposed is TDI. TDI is a transform in the time domain where the audio is first separated into multiple equal-length time windows, and the signal of each time window is reversed. The length of the time window (LTW) will be the only parameter for TDI. The second attack method is RPG which operates on the frequency domain. The signal is separated into equal-length time windows and then transformed to the frequency domain using FFT. Then, all frequency components are randomly rotated while preserving the original magnitude. The only parameter of the RPG is also the length of the time window (LTW). The third attack method is HFA, which adds high-frequency sine waves to the audio signal. There are two parameters that control the sine wave to be added into the audio, frequency (FREQ) and intensity (INT). The fourth attack is TS, which changes the tempo of the audio; the authors tried to speed up the audio while preserving its intelligibility to the ASR. The parameter that controls the speed of the audio is thus "tempo" (TEMPO).

5.2.2 Claims from Abdullah et al. [8]

Some conclusions of the parameters are pointed out by Abdullah et al. [8] in their paper: "The experimental results show that perturbation parameters, specifically window size (used for TDI and RPG), display three important properties. First, the smaller the window

size, the greater the audible distortion. Second, if an attack audio sample is successfully transcribed at a certain window size, then all attack audio samples that were generated with greater window sizes are also successfully transcribed. Third, no attack audio samples generated with window sizes of below 1.00 ms are correctly transcribed. " Besides, they also pointed out the possible parameter for their attack method "In our preliminary experiments we observed the TS factor of 150%, RPG or TDI window size of near 1.00 ms, and HFA of sine waves of frequency above 8000 Hz produced the ideal attack audio."

5.2.3 Reproducing Abdullah et al.'s work [8]

Using the conclusion provided by them, their four attack methods on 6 audio commands was evaluated separately. The results[2] of the reproduction test are presented in Table 5.2. For each attack method, the best perturbation parameter was applied, at the condition that the attack audio sample can be recognized by the Google speech recognition over-the-line. For TDI, the minimum LTW which is over 1 ms was applied. For RPG the minimum LTW which is above 1ms was applied. The RPG attack does not work for the command "open the door" no matter how the time window was increased; therefore it is marked as "✗". For HFA, the addition sine wave with the largest intensity at 11000 Hz was applied (randomly selected frequency which is above 8000 Hz, referring to the conclusion provided by the author "frequency above 8000 Hz"). The intensity parameter was set to 20000. The audio samples become annoying to human hearing but the ASR (refer to the audio clips in the corresponding table in our website) can still recognize them. I invite the reader to play the audio clip to verify that the audio is still intelligible despite how annoying the high-frequency pitch may be. For TS, the parameter provided by [8] (150%) was used and whether the command could be recognized by the ASR was tested. By testing each attack method of their work, some insight of their attack was provided.

I want to point out some of the flaws in the method proposed by Abdullah et al. [8].

[2]The audio samples for Table 5.2 are available to listen in this website (https://walterjohnson0.github.io/Compare_Practical.html).

Command	TDI	RPG	HFA		TS
	min LTW(ms)	min LTW(ms)	FREQ(Hz)	INT	TEMPO
Turn on airplane mode	3.00	20.83	11000	20000	1.5
Open the door	7.08	✗	11000	20000	1.5
Turn on the computer	2.08	6.25	11000	20000	1.5
Turn on the light	3.75	18.75	11000	20000	1.5
Call 911	3.91	10.41	11000	20000	1.5
Turn on Wi-Fi	3.00	10.41	11000	20000	1.5

Table 5.2: The reproduction of the work from Abdullah et al. [8]

The first flaw is that they did not specify the procedure or provide the code for the "perturbation engine" (PE). The way to search for the best parameter of the combination of TDI, RPG, HFA, TS was not clearly explained, which has a considerable impact when trying to reproduce their attacks. If the adversary is searching for all of the parameter in one shot they must generate $2^4 \times n(LTW_{TDI}) \times n(LTW_{RPG}) \times n(FREQ) \times n(INT) \times n(TEMPO)$ samples and pass all those samples into the ASR, in which $n(\cdot)$ is the number of the optional values for that parameter. This PE scheme will cause more than 200,000 queries on the ASR instead of 10 queries which were stated in Abdullah et al.'s [9] work.

One possible solution for the adversary to reduce the number of queries is searching for the best parameter on each attack method one by one instead of searching for all parameters altogether. If the adversary used this strategy, the number of queries could be reduced to $n(LTW_{TDI}) + n(LTW_{RPG}) + n(FREQ) + n(INT) + n(TEMPO)$ times. But the condition for this strategy is that there is no mutual interference on their four attack methods. This condition cannot be satisfied because adding/combining a second attack method will certainly reduce the intelligibility from the ASR or interfere with the parameter of other attack methods. One apparent mutual interference is between TDI, RPG, and TS. TDI and RPG use LTW as the parameter; If the adversary wants to overlay a TS on TDI or RPG, the time window for TDI or RPG will be modified because of time scalling. Another mutual inference is between HFA and TS; if the adversary does HFA first, and then do a TS, the frequency of the sine wave of HFA will be augmented. Therefore,

because of the mutual interference between their attack methods, the adversary cannot search for the best attack parameter one by one.

The second flaw is about the conclusion regarding TDI. The conclusion of "the smaller the window size the greater the distortion" was incorrect for the TDI. When searching for the best parameter for TDI, it did not show such features. Consider two extreme points. First, if the time window is super large, LTW equals the length of the whole audio signal, the audio will be totally reversed, which cannot be recognized by most ordinary people nor by the ASR. Second, if the time window is super small, the length becomes 1, the inversion of the time window of 1 does not modify any point of the signal, at this point there is no distortion on the audio. Therefore, for TDI, the conclusion should be on the opposite side, the smaller the window size, the less the distortion. This phenomenon is also demonstrated in our experiment; the smaller the window size, the perturbed audio sounds more like the original audio with high-frequency noise and easier being recognized by the ASR. To prove this, an additional experiment was performed searching for the maximum LTW that is lower than 1 ms for TDI. The result[3] is shown in Table 5.3 The attack audio samples with an LTW below those values can be recognized by the ASR.

Command	TDI
	max LTW(ms)
Turn on airplane mode	0.63
Open the door	0.63
Turn on the computer	0.52
Turn on the light	0.52
Call 911	0.41
Turn on WIFI	0.63

Table 5.3: The maximum LTW that is below 1 ms for TDI.

Parameter reusable test: One major difference between our work and Abdullah et al.'s [8] is the reuse of the parameter for one successful attack. In Abdullah's attack, they got

[3]The audio samples for Table 5.3 are available to listen in this website (https://walterjohnson0.github.io/TDI_LTW_Practical.html).

one parameter for one ASR system; for example, the time window length. Theoretically, in one successful attack, if they found out what is the time window length for one ASR system, they can apply the same parameter on the perturbation procedure for other commands.

The feasibility of the parameter reuse of one attack was tested. Theoretically, the parameter for their attack can be reusable, which can largely reduce the work when generating another command when attacking an ASR.

The command "call 911" was randomly picked. The parameter for attacking "call 911" was tried on other command attacking the same ASR. The result[4] is shown in Table 5.4. TDI and RPG are the methods that perform best, amongst the four, in reducing the intelligibility of the command. Therefore, the transferability of TDI and RPG can be considered as the key factor to judge transferability. The LTW of TDI is 3.91 ms for "call 911". We tested the same LTW-TDI for other commands and found that three of the two commands were correctly transcribed. The LTW of RPG is 10.41 ms for "call 911". Using the same parameter for the attack, two out of 5 commands were correctly transcribed. For HFA and TS, the perturbation parameter is more transferable only because the parameter is the same for all of the attacks.

Command	Attack Method			
	TDI	RPG	HFA	TS
Turn on airplane mode	✓	✗	✓	✓
Open the door	✗	✗	✓	✓
Turn on the computer	✓	✓	✓	✓
Turn on the light	✓	✗	✓	✓
Turn on Wi-Fi	✗	✓	✓	✓

Table 5.4: Test on the feasibility of parameter reuse of Abdullah et al.'s [8].

Practically, the attack has some transferability for TDI and RPG but is not stable. The attack on some commands cannot be transformed because they have different minimum LTW for TDI and RPG. And the best attack audio is not necessarily very unintelligible to humans.

[4]The audio samples for Table 5.4 are available to listen in this website (https://walterjohnson0.github.io/Reuse_Practical.html)

5.2.4 Comparing with the Intelligibility of the Attack Audio Samples

Short-time Fourier transform Abdullah et al. [8] proposed their hidden voice attack, attacking the signal processing stage of the ASR. Our attack makes more queries to craft one adversarial sample for the attack, but our attack can make the adversarial audio more unintelligible. Short-Time Fourier Transform (STFT) was used to compare our attack with the previous work from Abdullah et al.'s [8]. This comparison was conducted using the same audio command "Turn on the light" under an over-the-line attack. The targeted ASR model is Google speech recognition API [2]. Fig. 5.1 shows the power spectrum of the STFT for audio samples of the original command (Fig. 5.1, left), Abdullah et al.'s attack [8] (Fig. 5.1, middle) and our attack (Fig. 5.1, right). Random Phase Generation (RPG) was used to represent Abdullah et al.'s work [8] because, in our reproduction of their work, RPG gets the best disturbance on the audio among the four attack methods they proposed (at the condition that ASR recognizes the adversarial command). The sound starts from 1s and ends at around 2s. Abdullah's attack added perturbation in the power spectrum from 0Hz to 12kHz. Our attack added perturbation on all of the frequencies of the audio, covering up all the components from the original waveform. In the over-the-line attack, SegmentPerturb performs much better in reducing intelligibility using such a simple idea.

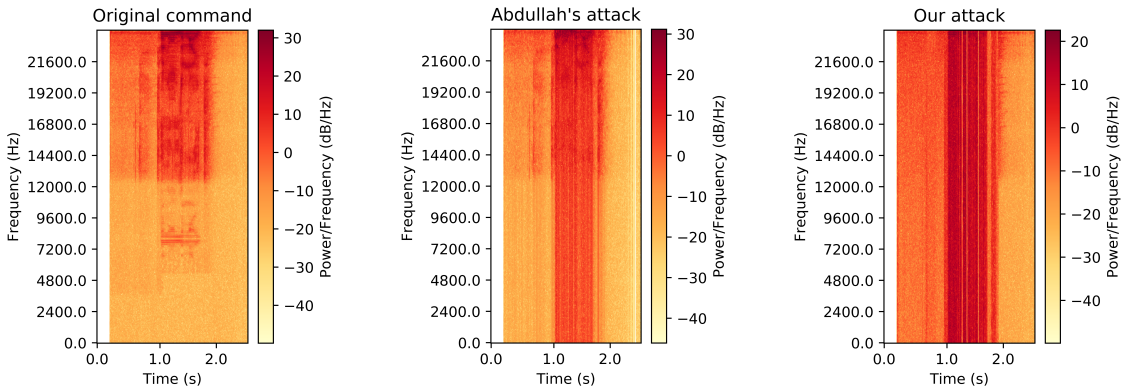


Figure 5.1: The STFT of the audio "Turn on the light".

Intelligibility test A similar intelligibility test was conducted on the audio sample from reproducing Abdullah et al.'s work [8]. Following the same procedure in Section 4.6, 100 HITs[5] were published in the Amazon Mechanical Turk, in which 50 HITs were testing TDI (pool D) and the other 50 HITs were testing RPG (pool E). The audio samples sequence for pool D is [Turn on airplane mode (TDI), Call 911 (TDI), Good morning (validation), Turn on the light (TDI), Open the door (TDI), Turn on Wi-Fi (TDI), How are you ("validation"), Turn on the computer (TDI)]. The audio samples sequence from pool E is [Turn on airplane mode (RPG), Call 911 (RPG), Good morning (validation), Turn on the light (RPG), Open the door (RPG), Turn on Wi-Fi (RPG), How are you ("validation"), Turn on the computer (RPG)]. The ratio of the participants who recognized the commands is shown in Table 5.5

Command	Abdullah et al.'s [8]				SegmentPerturb	
	TDI		RPG		Random Delete	
	1 st	2 nd	1 st	2 nd	1 st	2 nd
Turn on airplane mode	7%	16%	40%	47%	1%	2%
Open the door	93%	95%	91%	98%	27%	38%
Turn on the computer	80%	91%	86%	91%	42%	57%
Turn on the light	70%	82%	81%	91%	20%	32%
Call 911	100%	98%	91%	91%	15%	22%
Turn on Wi-Fi	91%	91%	83%	93%	45%	65%

Table 5.5: Audio intelligibility test for [8]

The table shows that both RPG and TDI do not totally destroy the intelligibility of the command. Most of the commands got as high as 90% of the participants' correct recognition, which suggests that the command is not very obfuscated. Compared with their perturbation method, SegmentPerturb did better in reducing the intelligibility of the voice command since the average of the first time comprehension is 25% for SegmentPerturb.

[5]The questionnaire is in Appendix A

5.3 SegmentPerturb as a Framework

SegmentPerturb is a framework for generating hidden voice commands by inquiring the ASR. The framework is starting from the ordinary voice command, separating the audio into equal-length time windows, and repeatedly applying perturbation on each segment while probing the ASR.

In this framework, the variable part is the MIPF. Other researchers can use different functions to substitute this part as long as the function satisfies the three characteristics of the MIPF. In this research, the baseline MIPF, "Random Delete", was proposed. This is the simplest MIPF yet works effectively in the over-the-line setting and performs well in over-the-air setting. However, it may not work consistently in the practical real-work attack in an over-the-air setting. Some potential reason for "Random Delete" to fail could be that the distortion of the sound transmission over-the-air is significant and the signal of the "Random Delete" cannot be physically reproduced by the vibration of the coil. The feasibility of using other MIPF in SegmentPertrub was demonstrated by testing on the "Random Delete in Spectrum". Therefore, other MIPFs can be designed to perform hidden voice attacks using the SegmentPerturb framework.

5.4 No Assumption on the Signal Processing Pipeline of ASR

Our attack does not involve assumptions on the signal processing stage. Previous works were mostly based on the assumption that speech recognition systems used some signal processing methods for feature extraction, especially MFCC or other spectrum-based methods as most of the ASR did use those signal processing methods. However with the rapid advances in the speech recognition models, there is a trend in modern ASR to avoid using those signal processing methods for feature extraction [23,43]. The assumption behind previous research on hidden voice attacks may not hold for emerging end-to-end models.

In the black-box attack experiment, previous works did make assumptions on the feature extraction of the speech recognition systems. Carlini et al.'s [13] research on the black-box attack was based on the assumption that the speech recognition system uses MFCC as the feature extraction method. In their black-box attack experiment, the audio mangler can generate an audio file that obtains similar MFCC features in the targeted ASR. Abdullah et al.'s [8] research claimed that they were performing their attack on a black box. However they also had one implicit assumption that the target ASR uses the time-frequency domain transform as a component for the feature extraction; otherwise, the theoretical support for their perturbation using RPG (Random Phase Generation) and TDI (Time Domain Inverse) is rather weak. Because the idea behind their work is that using their perturbation method on the waveform, the targeted ASR will get similar features after a time-frequency domain transform. Thus, the signal processing phase of the ASR in Abdullah et al.'s [8] work should include a time-frequency domain transform; otherwise, the adversarial audio will not generate a similar feature as the original audio.

Our hidden voice attack is based on a less restrictive assumption than previous works. No assumption was made on the signal processing or the time window separation. Therefore SegmentPerturb can be effective for all of the new emerging speech recognition systems. In SegmentPerturb, one MIPF and a way to inquire the ASR model can make an acceptable hidden voice attack.

5.5 Judging Whether the Voice Attacks Truly Fool the ASR

SegmentPerturb has good potential for judging the abnormal tolerance of ASR and whether voice attacks truly fool an ASR. ASRs are well-designed for recognizing vague speech. As such, the speech audio signal has overly redundant information in it. The ASR may use some non-critical features for recognition. Therefore, a well-designed voice attack may just

fall within the abnormal tolerance of the ASR instead of truly fooling the ASR.

Besides, typically, the abnormal tolerance for different commands is imbalanced. For example, Google speech recognition is more tolerant to the commands which the users use ordinarily and it would be easier to perform a voice attack on those commands. A similar phenomenon is also shown in other voice attack research. The voice attack research can get a better result on "Turn on the light", "Call 911" and "Play music". Because such standard commands are more tolerant to noise and therefore vulnerable to voice attacks. Using SegmentPerturb to probe those commands could be an effective way to show their difficulty to perform voice attacks; the APR can be used as the index of the difficulty. Besides, the tolerance for each segment in one audio command can also be revealed using SegmentPerturb. Therefore, SegmentPerturb could potentially become a comparison baseline for voice attacks. Other researchers may find out whether their voice attack samples truly fool the ASR or their perturbation audio samples were just within the abnormal tolerance by comparing to our work.

5.6 Defenses

5.6.1 Detect Electronic Sound

All of the over-the-air voice attacks on ASR have to use electronic speakers, therefore distinguishing the source of the voice as either from an electronic speaker or a human could be an effective way to defend against hidden voice attacks.

To prevent voice attacks, the "ASVspoof Challenge" began in 2015. In ASVspoof Lavrentyeva et al. [28] used Deep Neural Network to distinguish the replay spoof attack and they achieved an equal error rate (EER) of 6.73% which had a 72% relative improvement over the baseline system from ASVspoof 2017. Blue et al. [12] studied the difference between electronic and human voice. They studied the frequency domain characteristics at the sub-bass (20-80Hz) region and used this characteristic to distinguish electronic speakers

and the human voice. Their classification result gets a high true positive rate while keeping the false positive rate at a minimum. Ahmed et al. [10] further improve the method of detecting electronic sound; they proposed a fast and light electronic sound detection model, which only takes in 97 features.

But the defense method of detecting electronic sound may fail if the adversary carefully designs the adversarial audio or uses a high-quality audio device.

5.6.2 Using Other Interaction Methods to Confirm the High-Level Command

Some high-level commands are possible to be activated by one single sentence, which provides convenience for the adversaries. For example, one sentence commands like "Turn on airplane mode", "Call 911", "Order [item]" can activate certain behavior easily on the devices.

Therefore, using other interaction methods to confirm a high-level command is a good way for preventing easy activation of the high-level commands. For example, using a button to confirm the validity of the high-level command in the voice application in a mobile phone. If using a button is not possible for some devices, using other biometrics for authentication can also prevent the hidden voice attack.

Chapter 6

Conclusion

6.1 Limitations

6.1.1 Limited Transferability

SegmentPerturb is designed for attacking the specific target device, and the adversary can only perform this attack on the ASR models which they can query. For example, if the adversary wants to perform a hidden voice attack on Google Home, the adversary can perform the perturbation algorithm on a separate Google Home to obtain the perturbed audio file and then replay it to the target device. Alternatively, they could run the perturbation algorithm on the Google speech recognition API and then replay it over the air.

If the adversary cannot access the same ASR model, there is no theoretical support for this perturbation algorithm to succeed. Besides, there is no theoretical support for using the perturbed command for one ASR model to attack another model. The limitation in transferability restricts the application of this attack.

6.1.2 Attack Performance Depends on the Abnormal Tolerance of ASR

The success rate of our attack is highly related to the abnormal tolerance of the ASR model. If the model is more noise-tolerant, more segments in the waveform can be perturbed to a higher degree. In Fig. 4.1 and Fig. 4.2, Google and Azure are more noise-tolerant as they can recognize a meaningful command from strongly perturbed audio. For IBM speech recognition, even a small perturbation rate on the audio will make it generate the wrong output. The noise-resistant property of the ASR needs to be carefully designed for preventing this kind of attack. If the ASR system is more robust, it would be easier for the adversary to perform a hidden voice attack with a more unintelligible voice command.

6.1.3 Model Update of ASR Nullify the Attack Audio Samples for the Previous Version

This is a universal problem for all adversarial attacks on the ASR model. The commercial ASR are all experience a frequent update for improving the security or transcription accuracy. The attack on the ASR model is carefully searching for the perturbation on the audio which changed the output of the audio to a target malicious output. Commonly, those changes are delicate and only works for the specific model with the specific network parameter. These delicate perturbations in the attack sample can hardly adapt to the model with a tiny update in the network weight. This drawback also occurs in our attack, if the ASR has an update in the model, our attack audio sample may not necessarily workable for the next version of the ASR. Therefore, if the attackers want to attack the next version of ASR, they may need to craft a new version of the attack sample.

6.2 Future Work

The practicability of our attack was limited by the fact that voice assistants are comprehensive systems that receive commands and execute them instead of returning a transcription. Therefore, the ability of the adversaries to probe the ASR of the voice assistant was limited. Attacking a black-box voice assistant through the speaker is still a challenging task. In other words, it is hard to attack a voice control system for which the adversaries cannot probe the transcription. Therefore, future work could use zero-shot learning for attacking systems which the adversaries cannot inquire. And specific to the attack on ASR, transfer learning or model augmentation may also be a good research direction.

Another key problem for the practicability of the voice attack is how to overcome the audio distortion caused by the surroundings. Most of the state-of-the-art research on voice attacks was limited to the attack distance of two meters because of the distortion of the sound by the surroundings. To address the distortion from different surroundings may be a good direction to improve the attack distance.

Apart from the distortion of the surroundings, the difference of the speakers may also cause failure for the attack audio samples. However, most of the voice attack research only tests on a few different speakers but does not dig deep into how to address the problem of different speakers, as there are a large number of different speakers in the market. Therefore, how to address the problem of different speakers used by the users could also be a research direction to improve the robustness of the voice attack.

6.3 Conclusions

This thesis proposed a novel attack method for the black-box hidden voice attack on the modern ASR system. The goal of the attack is to generate noise-like attack audio samples that can be recognized by the ASR but cannot be recognized by the human. SegmentPerturb is tested using four commercial ASRs plus one mainstream smart home device.

In over-the-line attacks, SegmentPerturb can generate audio commands which can be recognized by commercial ASRs but cannot be recognized by a human. The result has shown that a smaller segment length would result in a more stable and effective attack. According to our observation, 20-30 ms would be a good segment length for SegmentPerturb. In over-the-air attacks, a series of practical experiments regarding SegmentPerturb were done. A visualization of the perturbation rate for every segment of the audio command was presented, showing that the segments which are deemed important from the view of the ASRs are not necessarily important from the view of the human. In a real scenario attack, the Google Home was placed at six representative positions in the bedroom, and most of the commands can be recognized at each position. The audio intelligibility test showed that only around 25% of participants can recognize the command when they first hear it. Therefore, the attack audio samples have some practicability and stealthiness in the real world.

Comparing with state-of-the-art hidden voice attacks, though SegmentPerturb requires more computational power, it gained more unintelligibility in the audio attack sample and had a longer attack distance. Besides, SegmentPerturb made no assumptions on the information of the target ASR.

SegmentPerturb provides a new way for hidden voice attacks. It exploits the vulnerability of the modern ASRs, that the ASR was so robust against the noise that even a noise-like audio sample crafted by selective deletion can still be recognized. This research implies that commercial ASR systems are vulnerable to voice attacks. Given the fact that the ASRs are widely deployed in most smart devices, little attention was paid to their vulnerability. This attack would not be successful if a minimum defense mechanism was applied, for example, detecting electronic sound, as pointed out in the discussion section. This finding may alert the designer of the modern commercial ASR and improve its security consequently. SegmentPerturb also raises doubts on the previous research of hidden voice attacks: did the previous research on hidden voice attacks truly fool ASR or the attack audio samples were just within the noise tolerance of the ASRs?

Bibliography

[1] Google cloud speech-to-text. https://cloud.google.com/speech-to-text. Accessed: 2020-06-07.

[2] Google speech recognition in pypi SpeechRecognition. https://pypi.org/project/SpeechRecognition/. Accessed: 2020-06-07.

[3] IBM speech-to-text. https://www.ibm.com/watson. Accessed: 2020-06-07.

[4] Kaldi platform. https://kaldi-asr.org/. Accessed: 2020-06-07.

[5] Microsoft Azure speech services. https://azure.microsoft.com/en-us/services/cognitive-services/speech-services/. Accessed: 2020-06-07.

[6] Voice assistant and smart speaker users 2020. https://www.emarketer.com/content/voice-assistant-and-smart-speaker-users-2020. Accessed: 2021-05-05.

[7] Wit.ai speech API. https://wit.ai/. Accessed: 2020-06-07.

[8] Abdullah, H., Garcia, W., Peeters, C., Traynor, P., Butler, K. R. B., and Wilson, J. Practical hidden voice attacks against speech and speaker recognition systems. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019* (2019), The Internet Society.

[9] Abdullah, H., Warren, K., Bindschaedler, V., Papernot, N., and Traynor, P. Sok: The faults in our asrs: An overview of attacks against automatic speech recognition and speaker identification systems. *arXiv e-prints* (2020), arXiv–2007.

[10] Ahmed, M. E., Kwak, I.-Y., Huh, J. H., Kim, I., Oh, T., and Kim, H. Void: A fast and light voice liveness detection system. In *29th {USENIX} Security Symposium ({USENIX} Security 20)* (2020), pp. 2685–2702.

[11] Alzantot, M., Balaji, B., and Srivastava, M. Did you hear that? adversarial examples against automatic speech recognition. *arXiv preprint arXiv:1801.00554* (2018).

[12] Blue, L., Vargas, L., and Traynor, P. Hello, is it me you're looking for?: Differentiating between human and electronic speakers for voice interface security. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks* (2018), ACM, pp. 123–133.

[13] Carlini, N., Mishra, P., Vaidya, T., Zhang, Y., Sherr, M., Shields, C., Wagner, D. A., and Zhou, W. Hidden voice commands. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016* (2016), T. Holz and S. Savage, Eds., USENIX Association, pp. 513–530.

[14] Carlini, N., and Wagner, D. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)* (2018), IEEE, pp. 1–7.

[15] Chang, S.-Y., and Morgan, N. Robust cnn-based speech recognition with gabor filter kernels. In *Fifteenth annual conference of the international speech communication association* (2014).

[16] Chen, T., Shangguan, L., Li, Z., and Jamieson, K. Metamorph: Injecting inaudible commands into over-the-air voice controlled systems. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020* (2020), The Internet Society.

[17] Chen, Y., Yuan, X., Zhang, J., Zhao, Y., Zhang, S., Chen, K., and Wang, X. Devil's whisper: A general approach for physical adversarial attacks against commercial black-box speech recognition devices. In *29th USENIX Security Symposium (USENIX Security 20)* (2020).

[18] Chiu, C.-C., Sainath, T. N., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., Kannan, A., Weiss, R. J., Rao, K., Gonina, E., et al. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), IEEE, pp. 4774–4778.

[19] Eddy, S. R. Hidden Markov models. *Current opinion in structural biology 6*, 3 (1996), 361–365.

[20] Graves, A., and Jaitly, N. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning* (2014), pp. 1764–1772.

[21] Graves, A., Jaitly, N., and Mohamed, A.-r. Hybrid speech recognition with deep bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding* (2013), IEEE, pp. 273–278.

[22] Graves, A., Mohamed, A.-R., and Hinton, G. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (2013), IEEE, pp. 6645–6649.

[23] Hannun, A. Y., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., and Ng, A. Y. Deep speech: Scaling up end-to-end speech recognition. *CoRR abs/1412.5567* (2014).

[24] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[25] Juang, B. H., and Rabiner, L. R. Hidden Markov models for speech recognition. *Technometrics 33*, 3 (1991), 251–272.

[26] Kwon, H., Kim, Y., Yoon, H., and Choi, D. Selective audio adversarial example in evasion attack on speech recognition system. *IEEE Transactions on Information Forensics and Security 15* (2020), 526–538.

[27] Lamere, P., Kwok, P., Gouvea, E., Raj, B., Singh, R., Walker, W., Warmuth, M., and Wolf, P. The CMU sphinx-4 speech recognition system. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2003), Hong Kong* (2003), vol. 1, pp. 2–5.

[28] Lavrentyeva, G., Novoselov, S., Malykh, E., Kozlov, A., Kudashev, O., and Shchemelinin, V. Audio replay attack detection with deep learning frameworks. In *Interspeech* (2017), pp. 82–86.

[29] Miao, Y., Gowayyed, M., and Metze, F. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (2015), IEEE, pp. 167–174.

[30] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al. The Kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding* (2011), no. CONF, IEEE Signal Processing Society.

[31] Qin, Y., Carlini, N., Cottrell, G., Goodfellow, I., and Raffel, C. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In *Proceedings of the 36th International Conference on Machine Learning* (Long Beach, California, USA, 09–15 Jun 2019), K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97 of *Proceedings of Machine Learning Research*, PMLR, pp. 5231–5240.

[32] Saon, G., Kuo, H. J., Rennie, S. J., and Picheny, M. The IBM 2015 English conversational telephone speech recognition system. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015* (2015), ISCA, pp. 3140–3144.

[33] Schönherr, L., Kohls, K., Zeiler, S., Holz, T., and Kolossa, D. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019* (2019), The Internet Society.

[34] Song, W., and Cai, J. End-to-end deep neural network for automatic speech recognition. *Standford CS224D Reports* (2015).

[35] Sugawara, T., Cyr, B., Rampazzi, S., Genkin, D., and Fu, K. Light commands: laser-based audio injection attacks on voice-controllable systems. In *29th USENIX Security Symposium USENIX Security 20)* (2020), pp. 2631–2648.

[36] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings* (2014), Y. Bengio and Y. LeCun, Eds.

[37] Vaidya, T., Zhang, Y., Sherr, M., and Shields, C. Cocaine noodles: exploiting the gap between human and machine speech recognition. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)* (2015).

[38] Xiong, W., Droppo, J., Huang, X., Seide, F., Seltzer, M., Stolcke, A., Yu, D., and Zweig, G. Achieving human parity in conversational speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing PP* (10 2016).

[39] Xiong, W., Wu, L., Alleva, F., Droppo, J., Huang, X., and Stolcke, A. The microsoft 2017 conversational speech recognition system. In *2018 IEEE Interna-*

tional Conference on Acoustics, Speech and Signal Processing (ICASSP) (2018), IEEE, pp. 5934–5938.

[40] Yan, Q., Liu, K., Zhou, Q., Guo, H., and Zhang, N. Surfingattack: Interactive hidden attack on voice assistants using ultrasonic guided waves. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020* (2020), The Internet Society.

[41] Yuan, X., Chen, Y., Zhao, Y., Long, Y., Liu, X., Chen, K., Zhang, S., Huang, H., Wang, X., and Gunter, C. A. Commandersong: A systematic approach for practical adversarial voice recognition. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018* (2018), W. Enck and A. P. Felt, Eds., USENIX Association, pp. 49–64.

[42] Zhang, G., Yan, C., Ji, X., Zhang, T., Zhang, T., and Xu, W. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), pp. 103–117.

[43] Zhang, Y., Chan, W., and Jaitly, N. Very deep convolutional networks for end-to-end speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2017), IEEE, pp. 4845–4849.

Appendix

A Audio Intelligibility Test

Instructions This experiment tests the intelligibility of up to 10 audio clips with noise. You will need to play each audio clip twice.

You should be aware that the audio clips are designed to be recognized by machines (e.g., voice control smart home devices) but not being recognized by humans. We do not include a "wake word" in the audio so that your smart home device will not be activated. To prevent accidental activation, please turn off all voice control devices nearby (within 2 meters).

Please click the Play button to listen to the audio clips and write down any words you can hear or understand in the first blank line. Then listen to the audio again, and write down what you can hear in the second blank line. Do not change the answer in the first line after you have played the audio for a second time.

For example, if the audio says "good morning everyone", but the first time you play it you only recognize "good", just input "good" in the first line. When you play the audio for the second time, if you recognize the whole phrase "good morning everyone", you can input "good morning everyone" in the second line. If you cannot recognize any word, please enter "N/A".

Repeat the same procedure for all audio clips.

Before starting the experiment, please play this audio file to make sure the volume on your computer is sufficiently loud. Once you adjust your audio please leave it like that for the whole experiment to make sure all audio clips are played at the same volume. [Volume test audio]

Input format: Please type in only lower cases with no punctuation. For example, if you hear "Good morning everyone", please input "good morning everyone".

Audio Experiment:

[Audio Link 1]:

[Input Box 1] [Input Box 2]

[Audio Link 2]:

[Input Box 1] [Input Box 2]

[Audio Link 3]:

[Input Box 1] [Input Box 2]

[Audio Link 4]:

[Input Box 1] [Input Box 2]

[Audio Link 5]:

[Input Box 1] [Input Box 2]

[Audio Link 6]:

[Input Box 1] [Input Box 2]

[Audio Link 7]:

[Input Box 1] [Input Box 2]

[Audio Link 8]:

[Input Box 1] [Input Box 2]

[Exit Question] If you cannot play the audio with the media player, please report the issue here.

[Input Box]

[Submit Button]