Practical Adversarial Attacks Against Black Box Speech Recognition Systems and Devices

by

Yuxuan Chen

Bachelor of Engineering
University of Electronic Science and Technology of China
Chengdu, China
2015

A dissertation
submitted to College of Engineering and Science at
Florida Institute of Technology
in partial fulfillment of the requirements
for the degree of

Doctorate of Philosophy
in
Computer Sciences

Melbourne, Florida
May, 2020
© Copyright 2020 Yuxuan Chen
All Rights Reserved

The author grants permission to make single copies.
We the undersigned committee
hereby approve the attached dissertation

Practical Adversarial Attacks Against Black Box Speech
Recognition Systems and Devices by Yuxuan Chen

William Allen, Ph.D.
Associate Professor
Department of Computer Engineering and Sciences
Committee Chair

Marius Silaghi, Ph.D.
Associate Professor
Department of Computer Engineering and Sciences

Shengzhi Zhang, Ph.D.
Assistant Professor
Department of Computer Engineering and Sciences

Fengkun Liu, Ph.D.
Assistant Professor
College of Business

Philip Bernhard, Ph.D.
Associate Professor and Department Head
Department of Computer Engineering and Sciences
Abstract

Title: Practical Adversarial Attacks Against Black Box Speech Recognition Systems and Devices

Author: Yuxuan Chen

Major Advisor: William Allen, Ph.D.

With the advance of speech recognition technologies, intelligent voice control devices such as Amazon Echo have become increasingly popular in our daily life. Currently, most state-of-the-art speech recognition systems are using neural networks to further improve the accuracy and efficacy of the system. Unfortunately, neural networks are vulnerable to adversarial examples: inputs specifically designed by an adversary to cause a neural network to misclassify them. Hence, it becomes imperative to understand the security implications of the speech recognition systems in the presence of such attacks. In this dissertation, we first introduce an effective audio adversarial attack towards one white box speech recognition system. Followed by this result, we further demonstrate another successful practical adversarial attack towards some commercial black box speech recognition systems and even devices like Google Home and Amazon Echo. We then discuss several methods to spread our adversarial samples by TV and radio signals. Finally, we turn to the defense for our attack and show possible
defense mechanisms to alleviate audio adversarial attack. In conclusion, this thesis shows that modern speech recognition systems and devices can be compromised by physical audio adversarial attacks, and also provides the preliminary results for further researches of how to design robust speech recognition systems to defend such attacks.
Table of Contents

Abstract ................................................................. iii
List of Figures ............................................................ ix
List of Tables ............................................................. xi
Acknowledgement ......................................................... xii

1 Introduction ............................................................ 1
   1.1 Overview of Speech Recognition Systems ......................... 4
   1.2 Security Concerns for ASR Systems ................................. 10
   1.3 Contributions and Summary ......................................... 12

2 Related Work .......................................................... 14
   2.1 Attacks for ASR Systems .......................................... 14
      2.1.1 Deep Learning Based Attacks ................................. 14
      2.1.2 Other Attacks ................................................ 27
      2.1.3 Defenses .................................................... 34
   2.2 Attacks Against Image Recognition Systems ..................... 37
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td><strong>White Box Audio Adversarial Attack</strong></td>
<td>43</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>43</td>
</tr>
<tr>
<td>3.2</td>
<td>Overview</td>
<td>46</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Motivation</td>
<td>47</td>
</tr>
<tr>
<td>3.2.2</td>
<td>The Philosophy of Designing This Attack</td>
<td>48</td>
</tr>
<tr>
<td>3.3</td>
<td>Approach</td>
<td>51</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Kaldi Speech Recognition Platform</td>
<td>52</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Gradient Descent to Craft Audio</td>
<td>55</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Practical Attack Over The Air</td>
<td>57</td>
</tr>
<tr>
<td>3.4</td>
<td>Evaluation</td>
<td>59</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Experiment Setup</td>
<td>60</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Effectiveness</td>
<td>61</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Towards the Transferability</td>
<td>65</td>
</tr>
<tr>
<td>3.4.4</td>
<td>Automated Spreading</td>
<td>67</td>
</tr>
<tr>
<td>3.4.5</td>
<td>Efficiency</td>
<td>68</td>
</tr>
<tr>
<td>3.5</td>
<td>Understanding the Attacks</td>
<td>70</td>
</tr>
<tr>
<td>3.6</td>
<td>Defense</td>
<td>73</td>
</tr>
<tr>
<td>4</td>
<td><strong>Black Box Audio Adversarial Attack</strong></td>
<td>76</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>76</td>
</tr>
<tr>
<td>4.2</td>
<td>Overview</td>
<td>80</td>
</tr>
</tbody>
</table>
5 Conclusions and Future Work

5.1 Conclusions ................................................................. 118

5.2 Future Work ............................................................... 119

References ................................................................. 119
List of Figures

1.1 Architecture of Automatic Speech Recognition System. . . . . . . . . . 4
1.2 An Example Waveform. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
1.3 Overlapped Frames. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
1.4 The Observed Sequences. . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
1.5 From frames to word. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
1.6 A High-level diagram. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
2.1 Architecture of Automatic Speech Recognition System. . . . . . . . . 16
2.2 Illustration of the Attack [39]. . . . . . . . . . . . . . . . . . . . . . . . 22
2.3 Workflow for Attack in [78]. . . . . . . . . . . . . . . . . . . . . . . . . 33
2.4 Architecture of Speakers. . . . . . . . . . . . . . . . . . . . . . . . . . . 35
2.5 Architecture of defense system. . . . . . . . . . . . . . . . . . . . . . . 36
2.6 Adversary Images in [95]. . . . . . . . . . . . . . . . . . . . . . . . . . 38
2.7 Adversarial Samples In Real World. . . . . . . . . . . . . . . . . . . . 42
3.1 Result of decoding “Echo”. . . . . . . . . . . . . . . . . . . . . . . . . 51
3.2 Steps of attack. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54
3.3 SNR impacts on correlation of the audios and the success rate of adversarial audios. .......................... 69
3.4 Explanation of Kaldi and human recognition of the audios. ............ 71
3.5 Audio turbulence defense. ........................................... 73
3.6 The results of audio turbulence defense. .......................... 74
3.7 Audio squeezing defense result. ................................. 75
4.1 Architecture of general adversarial attack against ASR API service and IVC devices. ................................. 84
4.2 Representative original song spectrum. ............................. 114
List of Tables

3.1 Relationship between transition-id and pdf-id. .......................... 52
3.2 WTA attack results. ......................................................... 60
3.3 WAA attack results. ......................................................... 62
3.4 Human comprehension of the WTA samples. ............................. 64
3.5 Human comprehension of the WAA samples. ............................. 65
3.6 Transferability from Kaldi to iFLYTEK. ................................. 67
4.1 The overall SRoC results on API services. ............................. 103
4.2 The overall SRoC results on IVC devices. ............................... 104
4.3 Transferability of the Devil’s Whispe AEs on Apple Siri. ............. 106
4.4 Results of the comparison tests with different approaches. .......... 108
4.5 Results of the human perception evaluation on Devil’s Whisper and
original song combined with TTS command. ............................. 112
Acknowledgement

I would like to thank my advisor Dr. William Allen for his guidance, advice, support, and help throughout my Ph.D. years at Florida Tech. Working with him, I felt more and more excited about my research work throughout these years.

This thesis would also not have been possible without Dr. Kai Chen who is working in Chinese Academy of Science. Dr. Chen gave me lots of useful advice and guidance for how to solve research problems and more importantly, how to conduct a good research project. I would also thank Dr. Chen for his time of reading and advising my papers.

Many thanks go to Dr. XiaoFeng Wang who is working in Indiana University, for our group discussions about our research projects and for his valuable suggestions. Furthermore, Dr. Wang gave me many insightful ideas on how to be a good researcher. I will definitely keep this in mind for my future research work.

I would also like to thank my committee members, Dr. Marius Silaghi, Dr. Fengkun Liu and Dr. Shengzhi Zhang, who have all offered invaluable guidance and advice during my years in Florida Tech. I especially thank Dr. Shengzhi Zhang. Dr. Zhang is very nice and always has time for me when I need help. I am also thankful for his time and effort in both reading my papers, proposals, dissertation and providing very useful suggestions for them.
Additionally, I would like to thank other lovely, smart and hard-working members in our research group, including Xuejing Yuan, Yue Zhao, Jiangshan Zhang, Aohui Wang and Yunhui Long, for their help, support, and encouragement.

I greatly appreciate anonymous reviewers’ insightful comments for my published papers, which helped me to improve our work. Specifically, I thank our shepherd for my USENIX paper, Professor Yongdae Kim, for his constructive feedback on this paper. We also thank Dohyun Kim and Taekkyung Oh in Professor Kim’s group, for their efforts to reproduce our results.

Finally, I thank my family, especially my parents, for their endless support and love. I dedicate this thesis to my parents: Hongwei Chen and Yifang Zhao.
CHAPTER 1

Introduction

With the fast development of speech recognition technology, the Automatic Speech Recognition (ASR) systems become increasingly popular these days due to the convenient and comfortable control over lots of functionalities and smart devices. Today, intelligent voice control (IVC) devices enabled with smart voice recognition skills like Google Home, Amazon Echo, Apple HomePod are already involved in our daily life which could execute the corresponding operations such as unlocking the doors of home or cars, making online purchase, sending messages, and etc. Also the availability of ASR services such as Google Cloud Speech-to-Text, Amazon Transcribe, Microsoft Bing Speech Service and IBM Speech to Text enables their users to conveniently integrate their APIs to control smart devices, conduct long-form audio transcription, text analysis, video analysis and etc.

Despite the immense popularity and convenience, ASR systems are exposed to various types of attacks [105, 104, 37, 39, 106, 65]. For example, some previous researches [106, 65] exploited the vulnerability of hardware which is used for pre-
processing module in ASR system workflow, then compromised the target ASR systems without victim’s awareness. Besides attacking the hardware part of the ASR systems, other researches [37, 28] explored the way that the feature extraction of the ASR systems could be manipulated. Carlini et al. [37] inverted MFCC [79] features of malicious commands back to audio and then added additional noise to obtain the attack audio samples uninterpreted to human beings but recognizable to ASR system and IVC device, e.g., Google Speech API and Google Assistant on smartphone. More recently, Abdullah et al. [28] developed four different perturbations to create the malicious audio samples, based on the fact that the original audio and the revised audio (with perturbations) share similar feature vectors after being transformed by acoustic feature extraction algorithms. Such attacks could bring severe impacts for victim’s property safety, i.e., the attackers can compromise the home IVC devices to open the garage door without victim’s awareness.

Recently, machine learning technology has been widely applied in ASR systems which vastly improves the accuracy and efficiency of speech recognition technology. However, a large amount of recent researches [37, 39, 82, 83, 84] showed machine learning models, especially deep neural network based models, are vulnerable to adversarial examples (AEs). That is, the original benign samples added with a small and unnoticeable perturbation, could be misclassified by machine learning models as other results. Specially, in speech recognition area, the adversarial samples may sound like the same as the original audio, but interpreted as other commands by ASR sys-
tems. For example, an adversary audio file which sounds like “good morning” by human beings, may be decoded as “open the door” by ASR systems. Some previous researches proved that it is completely feasible to generate such AEs for current ASR systems. Carlini et al. [39] have successfully attacked DeepSpeech (the open-source ASR model of Mozilla) using AEs, with the full knowledge of model parameters and architecture. Schonherr et al. [90] proposed that they could use psychoacoustic hiding to make imperceptible adversarial samples towards DNN-HMM based ASR system.

These above approaches demonstrate that the real-world ASR systems are vulnerable in a white-box model, when their internal parameters are exposed to the adversary. However, less clear is the security risks the commercial ASR systems such as Google Assistant, Microsoft Cortana, Amazon Echo and Apple Siri are facing, which would bring more real world impacts and influence millions of users. Although a recent study [96] exploited the feature extraction layer of DeepSpeech, treating it as a black box model, so far no success has been reported when it comes to generating AEs against the deep learning models behind commercial, close-source ASR systems, up to our knowledge.

Even more complicated is to attack physical IVC devices like Amazon Echo, since no longer can we receive real-time feedback from the target during the attack. Furthermore, when the successful AEs for target ASR systems are played over the air, the unknown environment and electrical noises would be introduced into the sounds, making the IVC devices more hard to interpret the sound as the target phrase.
In this section, we first introduce the workflow of a traditional speech recognition system, including how the system could transcribe the voice into the text result. Then we discuss that current speech recognition systems are vulnerable to several different types of attack. Finally, we summarize the key contributions of this work.

### 1.1 Overview of Speech Recognition Systems

ASR is a state-of-the-art technology which allows machines to understand human voice. Currently, there are a large amount of commercial ASR products like Amazon Echo, Apple Siri, etc; besides these, there are also some open-source ASR platforms like Kaldi [21], CMU Sphinx [8], HTK toolbox [15] and so on. Based on analyzing those open-source ASR systems, we can easily know what an ASR system’s workflow is and how it can transfer voice to text. Figure 1.1 gives us an overview of a traditional speech recognition system’s architecture, which includes two main procedures: feature extraction (containing pre-processing filtering) and model-based prediction (including both acoustic and language models).
After the raw audio is received from the microphone by the ASR system, it then passes the pre-processing filtering to filter out the frequencies which are out of human voice range and eliminate the voice segments below certain energy level. Next, the system will extract features from the audio for further analysis. The aim of this process is to convert the high-dimensional input data like raw audio to the low-dimensional information, so to ensure the machine could better handle it. Common acoustic feature extraction algorithms include Mel-Frequency Cepstral Coefficients (MFCC) [77], Linear Predictive Coefficient (LPC) [62], Perceptual Linear Predictive (PLP) [60], etc. Among them, MFCC is the most frequently used one in both open source toolkit and commercial products [81]. What’s more, after the acoustic feature of the raw audio is obtained, the ASR system will match the features with pre-trained acoustic models to predict the most possible words. Then according to the pre-trained language model, the ASR system will fix the language issue like grammar issue and commonly-used words. Note that after the feature extraction procedure, different ASR systems may have different models in the subsequent processes, and this may lead to various decoded results.

**The Workflow: From Voice to Text.** Next, a detailed example of how voice becomes text will be given:

First, we consider the time domain audio waveform as input. We know that sound is a kind of wave, and common audio file’s formats like mp3 and wmv are all compressed and must be converted to uncompressed pure waveform files, such as wav file
to handle. In addition to a file header stored in the wav file, it also has single point one by one to make up audio wave. The larger the sampling rate is, the more points are contained in each millisecond. What’s more, there are single-channel, double-channel and fourth-channel audio waveform files, and usually a single channel waveform is sufficient for ASR system to decode into text successfully. Figure 1.2 shows an example of a waveform.

Fig. 1.2 An Example Waveform.

Besides this, usually the waveform needs to do Voice Activity Detection (VAD) [52] process. This removes the silence segments in both head and tail part so to prevent the possible distortion in following steps. The waveform of the time domain must be divided into frames. That is, the waveform is cut into a small segment, and each segment is called a frame. Framing operation is usually implemented using a moving window function [89], and pre-emphasis [45]. Pre-emphasis could improve the power of the high-frequency part so to enhance the quality of the voice. After the frame division process, there is overlap between two frames, as shown in Figure 1.3.

In Figure 1.3, each frame is 25 milliseconds and there is 15 milliseconds overlap between two adjacent frames. The reason why we need frame division and overlap is
that, in a very short time range (10-30 milliseconds), the feature in this time slot can be considered stable and unchanged. The stability of signals is required for further analysis like Fourier Transformation [35], and the overlap could help ensure the smooth transition for each frame.

Now raw audio becomes many small segments. However, the waveform has almost no ability to exhibit features in the time domain, so it must be conducted transformation next. A common transformation method is to extract MFCC features and change waveform of each frame to a 12-dimensional vector. The 12 points are extracted based on the physiological characteristics of the human ear. It can be considered that these 12 points contain the context information of this frame of speech. This process is called acoustic feature extraction. In practical applications, this step is very complex and contains many other steps like differential, mean squared, dimension reducing etc.

At this point, the sound becomes a matrix of 12 lines (assuming the acoustic feature is 12 dimensions) and N columns. This matrix is called observed sequence, where N is the total number of frames. The observed sequence is shown in the figure 1.4 below.
In this figure, each frame is represented by a 12-dimensional vector. The color depth of the color block represents the magnitude of the vector value.

The next step is to introduce how to turn this matrix into text. First, two important concepts are given as below:

- **Phonemes**: The pronunciation of a word consists of phonemes. For English, a common phoneme set is Carnegie Mellon University Pronouncing Dictionary [67].

- **State**: A more detailed phonetic unit than the phoneme. Usually one phoneme consists of three states.

Now all we need to do are divided into three main steps: the first step is to identify the frame as a state (challenge). The second step is to combine the states into phonemes. The third step is to combine the phonemes into words. As Figure 1.5 shows.
In Figure 1.5, each small vertical bar represents one frame, several frames of speech correspond to one state, each three states are combined into one phoneme, and several phonemes are combined into one word. In other words, as long as you know which state corresponds to each frame of speech, the result of speech recognition will come out.

To solve the first step, we use Gaussian Mixed Model (GMM) [88] to determine which state is related to several frames. GMM is the linear combination of multiple Gaussian distribution functions, and theoretically GMM can fit any type of distribution [86]. For the second and third steps, we apply Hidden Markov Model (HMM) [49] to help us find the best possible result (word) for given states. HMM is a statistical model and can be considered as a sequence classifier. During this procedure, Viterbi algorithm [51] will be used to find global best path in the states network in HMM. Figure 1.6 shows a high-level framework of GMM-HMM based speech recognition system.
With the rapid growth of deep learning technology, DNN (Deep Neural Network) is used to replace the GMM model, since DNN could theoretically fit any function and the robustness is much stronger than GMM [27]. More recently, researchers came up with a new model called CTC (Connectionist Temporal Classification) [56]. This method is taking an audio sample as input, and transcribed sentences as output directly. It is less complex than DNN-HMM model since the whole mapping relation can be expressed in one neural network.

1.2 Security Concerns for ASR Systems

Nowadays, people are enjoying the convenience the ASR systems could bring and a large percentage of them are using ASR system enabled services frequently. Therefore, the security issue of ASR systems and devices remains a hot topic since it may
cause critical impact for victims. Many voice-enabled products hereby enhance their security check and remind users of potential malicious actions, however, some recent works [106, 65] show that even without direct access to the ASR systems and devices, an adversary could still compromise them and launch harmful actions for victims, without raise any awareness.

In Chapter 2, several representative works will be reviewed and compared. Depending on whether the adversary know the architecture and parameters of the ASR systems, there exists both white-box model like Kaldi [21] and black-box model like Amazon Echo. Also based on the attack methodology, the attacks can be categorized as several main subjects: hardware based, OS based and deep learning based attacks. A hardware level attack means it is not mainly towards modification for the original benign audio data, instead it usually explores other vulnerabilities related to the ASR systems such as the hardware for audio receiving process. Note since such attack will not explore inside truth of the ASR system normally, they often show strong robustness and transferability towards various black-box models. Compared with hardware based attacks, OS based attacks usually take advantage of operating system’s vulnerability to launch attacks against ASR systems. Finally, a deep learning level attack is mostly towards modification of benign audio samples, such attack usually cannot be understandable for human beings but can be interpreted as decided commands by machine. Differently with non-adversary attacks, only few of adversary attacks [42, 99] on ASR systems show transferability on black-box models, which limits their impact for real
world scenario. The main reason is that the complexity of ASR systems architecture significantly increases the difficulty for producing such adversary samples.

1.3 Contributions and Summary

The goal of this work is to explore the potential vulnerability of current speech recognition systems and devices under adversarial attacks. Therefore, as the preliminary step of the whole work, we first propose CommanderSong attack in chapter 3, a white box based audio adversarial attack towards open-source speech recognition toolkit Kaldi [21]. In this attack, we systematically embed target malicious commands into songs to attack speech recognition system by using gradient descent method. As demonstrated in this work, we show that white box speech recognition system could be vulnerable to such adversarial attack. Also, by adding random noise to further enhance the physical robustness of generated adversarial samples, we successfully launch the practical adversarial attack, which means the adversarial samples also survive the real world environment distortions. Chapter 3 introduces the in-depth explanation of our attack workflow and evaluation results for our attack.

Followed by CommanderSong, we then explore the possibility of adversarial attack for black box speech recognition systems, as such attack is more complicated and could bring severe impact to real world. In chapter 4, we propose the Devil’s Whis-
per attack, a black box based audio adversarial attack towards multiple commercial systems and devices like Google Home and Amazon Echo. In this attack, we show a novel strategy which is to combine a local large white box model with another surrogate model learned to approximate target black box model, so that the generated adversarial sample remain effective for target black box model. Meanwhile, we also make our sample robust to physical conditions to overcome real world challenges, then successfully launch physical attack for devices. Chapter 4 provides details of how we train the surrogate model and combine it with another white box model, plus the experimental results and evaluations.

Finally, we conclude the thesis and discuss some future research directions in chapter 5.
CHAPTER 2

Related Work

In this chapter, we will discuss related work for thesis topic, including audio attacks for speech recognition systems, adversarial attacks towards image recognition systems, adversarial attacks towards NLP (natural language processing) systems and defenses.

2.1 Attacks for ASR Systems

In this chapter, several previous researches towards deep learning based and other attacks for ASR systems will be reviewed and concluded.

2.1.1 Deep Learning Based Attacks

Currently, deep neural networks are employed for ASR systems to improve the accuracy and robustness, such as DeepSpeech [58]. However, a large amount of recent researches [44, 70, 36, 50, 95, 55, 99, 37, 39, 42] showed machine learning models, especially deep neural network based models, are vulnerable to adversary samples. That is, the original benign samples added with a small and unnoticeable perturbation,
could be misclassified by machine learning models. Specially, in speech recognition, the adversary samples may sound like the same as the original audio, but interpreted as other commands by ASR systems. For example, an adversary audio file which sounds like “Good Morning” by human beings, may be decoded as “unlock the door” by ASR systems. In this chapter, I reviewed works [99, 37, 39, 42] which are related to this topic.

**Attacks Against GMM-HMM Model.** Tavish et al. [99] proposed their research on exploring the difference between human and machine’s understanding toward voice sample and whether it can lead potential vulnerabilities. They proved it is possible and practical to build voice samples which are unintelligible to human being but can be interpreted as wanted commands by speech recognition systems.

In the threat model, the authors assumed the adversary could be physically close enough (around 3.5 meters) to the target speech recognition system or device and be able to play the malicious sound at a reasonable volume. This can ensure the target can hear the voice clearly. Besides this, it is required that the target system or device do not apply biometric authentication technology, since the voice which the adversary is going to play will not pass this check. Also, the victim need not be around nor using the device when the attack is initiated. Under this model, the goal of the adversary is to compromise target device’s voice recognition system, and make the system perform corresponding malicious actions, which will not raise any attention.
Fig. 2.1 Architecture of Automatic Speech Recognition System.

After establishing the threat model, one big challenge is that, different speech recognition systems vary in the ways to translate the audio to text, and these systems are proprietary with little knowledge about the inner mechanism and related parameters.

To overcome this, the authors mainly focus on the acoustic feature extraction process: regardless of how sophisticated the speech recognition systems are, feature extraction from the audio should be involved in one typical system, and the MFCC is most commonly used to represent acoustic features [64, 100]. Based on this, the authors build an audio mangler block, and the whole adversary workflow is as shown in Figure 2.1.

The audio mangler consists of two modules: feature extraction and inverse MFCC. First, in feature extraction module, the authors change several independent parameters which are required in MFCC extraction process. Second, the MFCC features are fed
into Inverse MFCC module, which will reconstruct the audio based on the MFCC features. Finally, the generated audio will be tested using the speech recognition system to ensure it can be interpreted as desired commands. If the audio couldn’t be decoded as wanted commands, the authors will return to the first step and tune the MFCC parameters again. After manually adjusting the range of parameter values which can make the produced audio recognizable by the machine as commands, the audio mangler is set up successfully.

The key factor of this approach is that, by passing the audio mangler block, the audio command still contains enough acoustic information which can be understandable by machine, while for human beings the mangled audio appears more like noises rather than actual commands. This is because in the Inverse MFCC process, the features that are not extracted in the last step, will not be applied in the reconstruction step, which makes the mangled audio more discernible for human but still sounds like commands for machine.

The limitations of this work are as follows: (i) For every single speech recognition system, this attack should repeat the procedure of adjusting MFCC parameters, since feature extraction settings may be varied for different systems. The lack of automatic and general approach makes this process time-consuming and inefficient. (ii) The attack scenario is limited since it requires both short attack distance and victim’s non-awareness of her device’s unauthenticated actions, unfortunately this is very rare in real world.
Followed by research [99], the community have the open question: what if the adversaries know detailed knowledge of the target voice recognition system? Could the attack be more unnoticeable by human? Researchers showed a more concealed adversarial voice command attack in [37].

In this work, the authors assumed the adversary could gain complete knowledge of the target voice recognition system, including algorithm for each step and corresponding parameters. The open-source CMU Sphinx speech recognition system [8] is served as white-box model in this research.

The CMU Sphinx speech recognition system is a popular open source platform used by some of apps and services. Recall the GMM-HMM model introduced before in Chapter 1, Sphinx system shares the similar workflow. It first runs the MFCC calculation to transfer the input audio information from high dimension to low dimension that the speech recognition system could process. Then it applies GMM to refer the probabilities that one given audio frame belongs to a certain phoneme. Finally, the HMM model is used to convert the phoneme to words.

By obtaining the additional information about parameters in MFCC extraction, the authors are now able to explore gradient descent method [33] to find a better solution than applying inverse MFCC procedure. Gradient descent is a general optimization method which is to find a best possible solution over a given space, and it can be set for arbitrary object function. Specifically, the object function in this scenario is expressed in Eq 2.1
\[ f(x) = (MFCC(x) - y)^2 \times z \] (2.1)

where \( x \) is every input frame, \( y \) is the target MFCC vector, and \( z \) serves as relative importance of each dimension which is set to \((1, 1, \ldots, 1)\) to be norm as the object.

Note although gradient descent method is not guaranteed to find global optimal solution but local optimal. The authors state it is sufficient for the aim. However, experiments show the results tested from adversarial video files retrieved from the gradient descent method make little progress compared with black-box attack. So the authors propose improved attack next.

Instead of targeting a better MFCC solution, now the authors explore further more and consider to use HMM model. They come up with two improvements over the previous method. First, since they have clear knowledge of GMM-HMM model for Sphinx system, now they can easily obtain a sequence of phonemes and further a set of HMM states for the target command (i.e., set breakpoint in the source code and analyze the output). Then they can try to look for a possible input which fits the HMM states. This way they can gain more freedom and even more choices, since theoretically different MFCC sequences are able to match with the same HMM states. For example, the English command “open the door” spoken by the American accent and Chinese accent may have different MFCC vectors, but for a robust speech recognition system, they shall be translated to same words, which means the same phonemes and
HMM states. Second, the authors attempt to decrease the frames per phoneme as few as possible, since the Sphinx system is not sensitive to the repeated HMM states but human being could notice that.

The samples generated from above idea work well if fed directly into Sphinx system, but the samples will not be decoded as desired commands if played over the air by speaker and received by Sphinx system. It is mainly because (i) the hardware part of speaker is not able to parse and play audios with large spikes like the samples, (ii) when the voice wave pass through the speaker and open air, the electrical noises from hardware circuits of speakers and environment noises from the air will add random distortion on the original voice signals, making it largely different with the wanted input to Sphinx system so that it cannot be translated as desired commands.

To overcome these challenges, the authors show three steps. First, they modify previous gradient descent algorithm to avoid large spikes produced in video samples. Second, they compare the waveform of original audio and record audio which has been played over the air. Generally, the simplified MFCC can be expressed as Eq 2.2 [37]

\[
MFCC(x) = C \log(B||Ax||^2)
\]  

(2.2)

where \(A, B, C\) are parameters for calculating MFCC. Now after the audio has been added with noise and distortion, the authors consider the new parameters \(A', B', C'\) and recorded audio \(y\), which implement Eq 2.3
Then they compute the new parameters $A'B'C'$ by least-square problem solving method. Finally, the authors did additional gradient descent for the recorded audio, the steps are: (i) before playing the samples over the air, the authors first run the gradient descent method to find the audio which is already close to target MFCC, (ii) then the authors played the generated audio over the air and recorded it, (iii) based on the recorded audio they obtained current MFCC, and they adjusted the target MFCC according to MFCC of recorded audio. This way they can retrieve more robust samples for the open air environment.

The results show the samples could fool Sphinx at success ratio over 90%. The authors also do a survey on Amazon Mechanical Turk [3] and show no workers can listen and transcribe the complete command words correctly. One clear and fatal limitation is that, compared with work [99] the generated adversarial samples are not transferable to any black-box model, which makes such attack only have limited impact.

**Attacks Against DNN Model.** Nicholas et al. [39] propose another novel attack which can compromise DeepSpeech [58], an open source CTC-based [56] speech recognition system. They prove that they are able to add a small and inaudible perturbation $\delta$ to the original waveform $x$, and the combined audio waveform $x + \delta$ can be interpreted as any desired command by DeepSpeech, as shown in Figure 2.2 [39].
In the threat model, the authors assume the attacker can gain the full knowledge of the target speech recognition system, which is DeepSpeech in this work. So given an original waveform $x$, regardless of music, voice or pure noise, and desired word sequence $y$, the aim of the adversary is to produce another audio waveform $x' = x + \delta$ and the transcript of $x'$ is same to $y$, but $x'$ and $x$ sound very similarly (defined later).

The authors apply Decibels ($dB$) to measure how similar $x'$ and $x$ is. Specifically, they compare distortion $\delta$ with the original waveform $x$ by equation 2.4

$$dB_x(\delta) = dB(\delta) - dB(x)$$  (2.4)
Since the added distortion is commonly quieter than original waveform \( x \), \( dB_x(\delta) \) will remain negative and the combined audio will be more similar as original waveform if the \( dB_x(\delta) \) absolute value is smaller.

The authors refer the previous work [34, 95] and consider the question of producing adversarial samples as optimization problem expressed as below: given a original audio \( x \), and desired target command \( t \), solve the equation 2.5:

\[
\begin{align*}
\text{minimize } & dB_x(\delta) \\
\text{such that } & C(x + \delta) = t \\
& x + \delta \in [-M, M]
\end{align*}
\] (2.5)

Here \( \delta \) is the small perturbation added to the audio, \( dB_x(\delta) \) represents distortion metrics defined before, \( C(.) \) means the transcriptions of DeepSpeech system, and \( M \) defines maximum possible value. The equation states that in all possible value zone, they wish to look for some certain perturbation delta, so that the combined audio could be interpreted as wanted phrase. In other words, \( x + \delta \) is formally closest audio to \( x \) classified as \( t \) by \( C \).

In [34, 95], researchers considered this question by using a box-constrained L-BFGS [72] and line-search [57] algorithm and reformulated the above optimization problem as Eq 2.6:
\[
\minimize dB_x(\delta) + c \cdot l(x + \delta, t)
\]

(2.6)

where \( l(x', t) \leq 0 \iff C(x') = t \)

The parameter \( c \) restrains how close the adversarial sample is with original example and strength of the adversarial samples. The \( l(.) \) is the loss function in the adversarial sample training process and needs to be defined by attackers. In image fields, the building of such loss function is simple: \( f(y, x) \) can perform the probability of input image \( x \) has result \( y \), however for audio recognition, it is non-trivial since audio signal has more complex decoding process than image.

To overcome this challenge, the authors recall how CTC-based recognition systems define the loss function, since when compared to traditional speech recognition system like GMM-HMM model and DNN-HMM model, CTC model has more similar process with image recognition systems: the input of CTC is an audio sample of one person speaking and the labels are transcript results. According to [56], the loss function in CTC is as Eq 2.7:

\[
CTC_{Loss} (f(x), p) = -\log Pr(p|f(x)).
\]

(2.7)

which means the negative log likelihood of a given phrase \( p \) under the distribution \( f(x) \). In simple words, it would maximize the probability it assigns to the right answer.
Now, they make loss function in the adversary model to be Eq 2.8:

\[ l(x', t) = CTC_{\text{Loss}} (x', t) \]  \hspace{1cm} (2.8)

Note that different with 2.6, this time only one direction remains true (\( l \Rightarrow C \)), since there may exist many solutions which can make the sample adversarial, but not all solutions are minimally perturbed.

After obtaining the loss function, now the new challenge is that, under the maximum distortion metric as defined before in Eq 2.5, usually the solution will not converge to a final value [38]. So the authors apply the trick in [38] and establish the following formulation Eq 2.9:

\[
\text{minimize } |\delta|^2 + c \cdot l(x + \delta, t) \\
\text{such that } dB_x(\delta) \leq \tau 
\]  \hspace{1cm} (2.9)

This restraints \( \delta \) to have distortion maximum at \( \tau \). In experiments, the authors give \( \tau \) a large value at start, after obtaining solution \( \delta \) then they reduce \( \tau \) to see if such \( \delta \) still exist, until \( \tau \) cannot be lower anymore.

The results show the attacker could produce targeted audio adversarial samples with 100% success rate with a mean distortion -31 dB. While they can not make the adversarial samples useful after being played over the air and the samples do not
show any transferability towards other speech recognition models, this finding is still remarkable since it remains the first targeted adversarial attack with a very high success rate (100%). Previous successful targeted work [106, 37] either needs some expensive hardware or requires constructing a new audio.

Besides the work [39], Moustapha et. al [42] and Gong et. al [53] also take advantage of an intriguing property of DNN by generating malicious samples that are very similar with original benign audios by conducting the optimization method. In [53] the authors can produces such adversarial samples which could make cutting-edge neural network misclassify the gender and identity of speaker. In [42], the authors show the generated adversarial samples can even fool the commercial black-box ASR model such as Google Now, despite the samples are non-targeted so the impact is limited.

Though the adversarial examples of attacks for DNN model are more stealthy than previous samples for GMM-HMM model [37], these methods all have a clear limitation: since the perturbations are usually minor, when the samples are played over the air and captured by the ASR enabled devices like Amazon Echo, they will be highly impacted by environment and electromechanical noises hence fail to be decoded to desired text. In the work [39], [42] and [53], the authors all stated their attack cannot work for such scenario. This phenomenon significantly decreases the impact such attacks can bring, since in real world people will use ASR devices this way. Also, To the best of my knowledge, there does not exists any targeted DNN-based adversarial samples which could fool the black-box model under the real world scenario. Pre-
vious successful work [106, 99] either exploit hardware level vulnerability or remain non-adversarial. So this question is left for the future work in the community.

2.1.2 Other Attacks

Besides the above deep learning based attacks, there also exists other types like hardware based attacks, simple voice replay attacks, OS (Operating Systems) based attacks and speech misinterpretation attacks. Hardware based attacks are normally not exploiting ASR system’s inside vulnerability, instead by using the characteristics of the hardware (e.g., the analog-digital converter) to make the attacks stealthy. In this section, two papers [65] and [106] about hardware based attacks will be reviewed. A voice replay attack means the adversary could replay a previously recorded voice from the victim, so to force the ASR systems launch malicious actions, as proved in [102, 71]. This type of attack is easily detected by victim so in real world scenario the impact is very limited. The OS level attack is more tricky, the attacker could exploit the vulnerability in OS such as Android to make the attack self-triggered and more imperceptible. For example, in [47], the authors demonstrated that, through permission bypassing attack in Android smart phones, voice commands could be played using apps with zero permissions. Besides above attacks, speech misinterpretation attacks also exist since model based prediction module in ASR systems consists of both audio and language model and speech squatting attack could compromise language model instead of audio model.
Automatic Speaker Verification (ASV) system is also raising people’s interest recent years, since voice remains unique biometric characteristic for private authentication [46]. However, sharing with the similar architecture with ASR systems, ASV systems are also facing various powerful and harmful attacks recently as given in [78, 29, 101, 94]. In this section, the work [78] towards attacking ASV systems will be reviewed.

**IEMI Threats: Remote Command Injection on Modern Smartphones.** Since most speech recognition systems in physical world need both two parts: software part for feature extraction, model prediction, language and grammar check; hardware part for signal receiving and processing and etc, so people may ask is there any existing hardware related weakness which we could utilize to compromise speech systems?

The answer is obviously YES. Chaouki Kasmi and Jose Lopes Esteves propose a novel attack for silent remote voice command injection on modern smartphones in their paper [65]. They can broadcast the voice commands signal remotely by SDR (software defined radio) and a proper amplifier. Then the cellphone with headphone plugged in will receive the signal and then consider voice it from microphone so to decode them later.

In the threat model, the authors assume the adversary do not have any direct access to the cellphone, but she can deploy the SDR devices less than 10 meters away from the victim’s cellphone. Also, the cellphone must have one headphone plugged into the jack and the voice assistant app like Siri and Google Now must be enabled. Finally,
though the authors do not mention in the paper, the owner of the cellphone must not be around or aware of any abnormal of the phone when attack is launching the attack.

The key concept for the implementation of this attack is that, the headphone cord is served as antenna when plugged into the phone. This antenna can be considered as a front-door coupling [32] interface which could transfer electromagnetic signals into electrical signals. The electrical signals will be considered as the incoming audio input from microphone by cellphone. Interested readers can refer to [97] for more details about intentional electromagnetic interferences (IEMI) and relevant vulnerabilities.

Despite this novel attack remains stealthy and silent for the victim, and may cause severe impact due to the various functionalities of ASR-enables apps in cellphone, the limitations of this attack are as below: (i) although the adversary is able to make the whole injection silent and will not raise any awareness from victim, the SDR equipments must be set up as close as 1.20 meters away from cell phones, which is too close to raise other notice, (ii) it requires the cell phone plugged with one headphone to launching the attack, however, many users will plug out the headphone if they are not using it.

**DolphinAttack: Inaudible Voice Commands.** Similar with the above work, Zhang et al. [106] in their work named DolphinAttack, show that the adversary is able to inject totally inaudible voice commands to cutting-edge and widely-used speech recognition systems like Siri, Google Now, Alexa and etc. Then the attacker could launch kinds of
malicious actions for those systems. The main difference with the work [65] is that, in DolphinAttack, there is no need to plug the headphone into the cellphone.

In their threat model, the authors assumed that (i) the adversary is not able to directly access the target device; (ii) the victim will not interact with the device at the moment when adversary is launching the attack, also the response of the device will not raise any attention for victim otherwise she can immediately force it to stop; (iii) the adversary could acquire appropriate speaker and modulator, in order to complete the attack.

Recall that the main aim of this attack is how to make voice commands inaudible. As one general knowledge, we people could hear the sound with frequency less than 20 kHz. It is obvious that if we can exploit ultrasound channel ($f > 20kHz$) as command carrier, the voice may sound inaudible for human beings. Unfortunately, most audio-capable devices are applying LPF (Low-Pass Filters) to eliminate signals with frequency larger than 20 $kHz$.

In electronics fields, electric elements like amplifiers are supposed to be linear, but in reality, due to the nonlinearity property of transistors, electric components exhibit nonlinearity. The authors did some experiment and proved the microphone used for receiving voice signals has nonlinearity effect. In details, let the attack commands come with central frequency $f_m$, and the ultrasonic carrier waveform with central frequency $f_c$. After taking the Fourier transform, the output signal would contain components with central frequency $f_c$, $f_m$, $f_c - f_m$, $f_c + f_m$ and etc. Since all frequencies except
$f_m$ are larger than 20 $kHz$, the $f_m$ frequency component will stay but others will be removed by a LPF.

After investigating the vulnerabilities of the hardware and proving the attack’s feasibility, the authors made use of TTS (Text to Speech engine) to generate corresponding voice commands and amplitude modulation (AM) to modulate voice commands on ultrasonic carriers. Then they chose proper equipments as signal generator, modulator and speaker. The results show the attack could successfully compromise 16 devices and 7 state-of-the-art speech recognition systems including Apple Siri, Google Now, Microsoft Cortana, Huawei Hi Voice and etc.

The limitations of DolphinAttack are as follows: (i) the attack requires the adversary to use high-end ultrasonic signal generators, however, such generators are normally very expensive and could raise victim’s awareness if employed around her house or office. (ii) under the author’s threat models, as long as the victim notice the abnormal of the ASR systems or devices, she could immediately force the malicious actions stop. In real world, this could happen frequently.

**Speech Misinterpretation Attack.** Recently, third-party applications and skills for IVC systems become increasingly popular, while the lack of proper authentication raises security and privacy concerns. Previous studies show third-party applications are facing misinterpretation attacks. Unlike previous works aiming at generating audio adversarial samples [37, 39, 105], such attack usually exploits intrinsic error within the opaque natural language processing layer of speech-recognition systems.
Kumar et al. [69] present an empirical analysis of the interpretation errors on Amazon Alexa, and demonstrate the adversary can launch a new type of skill squatting attack. Zhang et al. [107] report a similar attack, which utilizes a malicious skill with the similarly pronounced name to impersonate a benign skill. Zhang et al. [108] developed a linguistic-guided fuzzing tool in an attempt to systematically discover such attacks.

**Attacks Against ASV System: Stealing Voices to Fool Humans and Machines.**

In [78], the authors showed that, after obtaining very limited voice samples spoken from victim, the adversary is able to produce any desired audio commands, which is too similar to be distinguished by both machine and human. Specifically, the key part is the using of state-of-the-art voice morphing techniques to can convert the attacker’s voice to victim’s voice. The results show the generated voice could fool both machine-based and human-based voice authentication systems, at a high success rate.

In the threat model, the attacker could gain a few voice samples of the victim for further training process in voice morphing tools. This is not unfeasible in modern world since people always leave segments of their voices in many different environments and scenarios like restaurant, library, shopping mall and etc. What’s more, the attacker should have a direct physical access to devices with voice authentication systems, so he could launch the attack.

Figure 2.3 [78] shows the whole workflow of the attacking process. Firstly, the attacker should collect voice samples $O_T = (t_1, t_2, ..., t_n)$ of the victim’s speech, the sample length is approx 6-8 minutes and there is no need to use high-end recording
equipments. Secondly, the attacker speaks the same voice as the victim’s speech $O_T$, labeled as $O_S = (s_1, s_2, ..., s_n)$, then $O_S$ and $O_T$ are as input to voice morphing tool to train the model $M = \mu(O_S, O_T)$ to simulate the victim’s voice. In this work, the morphing engine is CMU Festvox Voice Conversion System. It applies acoustic-to-articulatory inversion technology [98] to establish the relationship between sound signals and articulatory movements for human beings, so after being trained by both victim’s and attacker’s voice samples, this engine is able to learn and build speaking models for both victim and attacker and then can generate the corresponding speech in the target’s voice based on given victim’s speech. Finally, when the attacker speaks arbitrary sentence $A = (a_1, a_2, ..., a_m)$, the morphing engine is able to convert it into the victim’s voice as $f_T = M(A) = (f_1, f_2, ..., f_m)$. The generated voice $f_t$ will be tested for both machine-based and human-based speaker verification systems.

The authors tested the voice on Spear verification toolbox developed by Khoury et al. [66] for machine-based authentication and Amazon Turk for human-based authen-
tication. The success rate is about 80-90% against Spear system and about 50% of the cases, human verifiers were fooled by morphed samples.

The main limitations of this attacks are: (i) the authors did not test the samples towards other ASV systems, so the robustness of such attack remains unknown. (ii) Although as the authors stated, they only need very limited audio samples for attack, 6-8 minutes are still a little long and hard to obtain.

2.1.3 Defenses

With the rapid emerging of state-of-the-art attacks for both ASR ans ASV systems, it is in high demand for a strong and robust defense mechanism for those attacks. Many previous work were paying attention to the defense method. An idea for detect the machine learning based attacks is adversarial training, which means training a special deep learning model for detecting adversary samples. Previous researches proved this idea is efficient for defending machine learning based attacks [37, 106]. However, this defense requires the knowledge about how adversary generate their samples, in real scenario this is not happening frequently. Other defenses [37, 78] are towards using speaker verification systems but they are easily fooled by adversary samples. In this section, a robust and efficient way of defense for various types of attacks will be reviewed.

Since for both machine learning based and hardware level attacks, the adversary needs to apply the speaker [37] or signal generator [106] to complete the full attack.
So obviously, if we can detect the difference between voice from electronic devices like speakers and human beings, many powerful attacks such as [106, 37, 99] could be defended since they all require using such electronic devices.

In [40], the authors proposed such a defense method and results showed it is highly effective for detecting voice impersonating attack on smartphones.

The key idea of this defense is that, most conventional speakers contain a permanent magnet, a coil as electromagnet and a cone for converting electronic signal to voice signal and broadcast it over the air [87], as shown in Figure 2.4 [40]. So it is not surprised that when the speaker is working, the voice signal would contain magnetic field which we human beings do not have. The authors noticed this finding and designed the defense architecture as shown in Figure 2.5 [40].

The architecture contains four parts: sound source distance verification, sound field verification, loudspeaker detection, and speaker identity verification components.
The sound source distance verification part is to ensure the detection system is close enough to the sound source, in order to guarantee the functionality of the magnetometer. In the experiment, the authors showed this distance should be around 6 cm. The second part, sound field verification is another issuance for small speakers which did usually not produce enough magnetic field in the voice so could bypass the magnetometer check. This part is mainly to check such signals to ensure it can get negative results. The third part is the main part for magnet filed detection. Finally, the fourth part is for human voice verification, and the authors here applied the open-source Bob Spear verification tool [66] to implement the architecture.

![Architecture of defense system.](image)

Fig. 2.5 Architecture of defense system.

Although this defense is powerful and successful for detecting lots of attacks [37, 99, 42] introduced before, the limitations still exist as follows:

- The defense only works up to 10cm, which is less than the usual speaker’s range.
• The defense is mainly towards traditional speakers which have magnetic components, however not effective for other non-traditional or signal generators which do not have required magnetic property.

2.2 Attacks Against Image Recognition Systems

Prior to the researches for attacks on speech recognition systems and speech authentication systems, there has been substantial work on exploiting vulnerabilities on machine learning systems [44, 70, 36, 50, 95, 55, 99, 37, 39, 42] and image recognition systems [73, 59, 80, 75, 76, 63, 55, 95, 30], especially adversary samples towards these systems. In this chapter, several meaningful and important works related to this topic will be reviewed.

Christian Szegedy et. al [95] firstly showed current deep learning model which contains neural networks like CNN [92] can be vulnerable to adversarial sample, which is the original benign sample added with a very small and unnoticeable perturbation but can be misclassified by the deep learning model. Figure 2.6 gives us a straightforward look about how such adversary sample looks like. The left is a correctly predicted sample, the center is the perturbation that is difference between correct image and adversary image, and the right is the image predicted incorrectly by the neural network. All images in the right are predicted to be an ostrich with high confidence by the AlexNet [68] image recognition system.
The authors also proposed one method to produce such adversarial samples: given original image $x$ and target label $l$, they can transfer this question to a box-constrained optimization problem:

- Minimize $||r||^2$ subject to:
  
  1. $f(x + r) = l$
  2. $x + r \in [0, 1]^m$

Here, $r$ represents the distortion and $f$ is the mapping between the input and the output. The problem can be expressed as: with a constrained value zone for $x + r$, looking for a minimum value $r$, to make $x + r$ can be interpreted as $l$. Then the authors applied a box-constrained L-BFGS algorithm [72] to solve the computation of $r$, then the solution now is:

- Minimize $c|r| + \text{loss}_f(x + r, l)$ subject to $x + r \in [0, 1]^m$
which fulfillments $f(x + r) = l$.

As the authors stated, such adversary sample $x + r$ has the following properties: first, the distortion $r$ remains totally discernible and unrecognizable for human beings, nonetheless the sample $x + r$ could be interpreted as another object by machine. Second, such adversary sample could not only compromise the target deep learning model, but also be effective towards other models. In other words, they exhibit transferability. Similarity, the work [80] also proved this and such samples may also be extended to some commercial black-box models.

The finding in [95] is revolutionary and significant, because before that time, community always believed a well-trained neural network could perform high success rate on classification. However, now the neural networks can be easily fooled by adversary examples. This would raise severe security concerns because a large amount of cutting-edge technologies like face recognition, automatic driving, scam email filter and etc are applying neural networks to help improve accuracy and efficiency. So such adversarial sample is potentially harmful for victim’s property, safety and security on many fields in the future.

Followed by work [95], Ian Goodfellow et al. [55] gave the reason why existing neural networks show vulnerabilities in previous research. Unlike many scientists thought that extreme nonlinearity of the deep neural network is the main cause of such vulnerability, they considered the linearity property of the deep neural network is more than enough to lead this vulnerability.
This explanation could be understood in this simplified way: since the input feature only has limited precision, the classifier is not able to discriminate the original input sample and the sample added with the perturbation which is smaller than precision to be discarded by classifier. Next, this small distortion will be amplified by the dimension of the weight vector. So as long as the dimensionality of the input sample is sufficient, the linear model can have adversarial examples.

The authors also proposed a quick and simple way to generate adversarial samples for neural network, named Fast Gradient Sign Method (FGSM). This method is later extensively used in various adversarial machine learning researches and proved to be efficient and useful.

With the theoretical knowledge support from above work, a large amount of related researches [73, 59, 75, 76, 63, 30] were conducted in last years.

In work [70, 30], the researchers showed the adversary samples are not only effective in virtual scenarios (fed directly into API of image recognition systems), but also powerful in physical world scenarios. The authors proved that even the adversary images are printed and photographed by camera, they can still fool classifier successfully. This finding is meaningful since now the adversary samples show strong robustness in real world, so the attacks for real world applications could be launched to cause severe consequences. For example, in [91]. The authors gave us evidence that the eyes with adversary perturbation could make state of the art face recognition systems misclassified.
Another real world scenario with dangerous impact is autonomous vehicles and automatic driving technology, since the speed and direction are fully decided by the machine. In [74], researchers built adversary road sign images by using FGSM [55] and L-BFGS [72] algorithms and found in real world driving conditions these samples are not adversary any more, since the angle and distance between the car camera and road sign will change frequently, which would significantly decrease the success rate of adversary samples.

However in work [50], the authors proved that it is feasible to build such adversary road sign images to overcome the real world challenges and still be effective in the driving condition. They came up with two different attacking methods: subtle perturbations and camouflage perturbations. For subtle perturbations, the attacker would print the whole adversary road sign image and cover it onto the original road sign, while for camouflage perturbations, attacker would only add some graffiti arts on the original road sign. Figure 2.7 [50] shows a successful camouflage perturbations attack for stop sign, the stop sign will be recognized as speed limit sign by the trained classifier.

To sum up, adversary attacks on neural network based image recognition systems are extensively conducted by the community recent years, while such research for speech recognition systems remains few. To my knowledge, the main reason is that, for speech recognition systems, the audio signal contains continuous information so the feature extraction process is more complex than image systems, specifically, this
Fig. 2.7 Adversarial Samples In Real World.

may lead to increased difficulty for the loss function design when running the gradient descent algorithm, which is a popular method for producing adversary image.
CHAPTER 3

White Box Audio Adversarial Attack

In this chapter, the details of CommanderSong attack will be given, including motivations, threat model, approaches and evaluations.

3.1 Introduction

Intelligent voice control (IVC) has been widely used in human-computer interaction, such as Amazon Alexa [2], Google Assistant [11], Apple Siri [5], Microsoft Cortana [24] and iFLYTEK [19]. Running the state-of-the-art ASR techniques, these systems can effectively interpret natural voice commands and execute the corresponding operations such as unlocking the doors of home or cars, making online purchase, sending messages, and etc. This has been made possible by recent progress in machine learning, deep learning [61] in particular, which vastly improves the accuracy of speech recognition. In the meantime, these deep learning techniques are known to be vulnerable to adversarial perturbations [70, 36, 50, 44, 34, 95, 55, 83]. Hence, it
becomes imperative to understand the security implications of the ASR systems in the presence of such attacks.

**Threats to ASR** Prior research shows that carefully-crafted perturbations, even a small amount, could cause a machine learning classifier to misbehave in an unexpected way. Although such adversarial learning has been extensively studied in image recognition, little has been done in speech recognition, potentially due to the new challenge in this domain: unlike adversarial images, which include the perturbations of less noticeable background pixels, changes to voice commands often introduce noise that a modern ASR system is designed to filter out and therefore cannot be easily misled.

Indeed, a recent attack on ASR utilizes noise-like hidden voice command [37], but the white box attack is based on a traditional speech recognition system that uses a Gaussian Mixture Model (GMM), not the DNN behind today’s ASR systems. Another attack transmits inaudible commands through ultrasonic sound [106], but it exploits microphone hardware vulnerabilities instead of the weaknesses of the DNN. Moreover, an attack device, e.g., an ultrasonic transducer or speaker, needs to be placed close to the target ASR system. So far little success has been reported in generating “adversarial sound” that practically fools deep learning technique but remains inconspicuous to human ears, and meanwhile allows it to be played from the remote (e.g., through YouTube) to attack a large number of ASR systems.

To find *practical* adversarial sound, a few technical challenges need to be addressed: (C1) the adversarial audio sample is expected to be effective in a complicated,
real-world audible environment, in the presence of electronic noise from speaker and other noises; (C2) it should be stealthy, unnoticeable to ordinary users; (C3) impactful adversarial sound should be remotely deliverable and can be played by popular devices from online sources, which can affect a large number of IVC devices. All these challenges have been found in our research to be completely addressable, indicating that the threat of audio adversarial learning is indeed realistic.

**CommanderSong.** More specifically, in this paper, we report a practical and systematic adversarial attack on real world speech recognition systems. Our attack can automatically embed a set of commands into a (randomly selected) song, to spread to a large amount of audience (addressing C3). This revised song, which we call CommanderSong, can sound completely normal to ordinary users, but will be interpreted as commands by ASR, leading to the attacks on real-world IVC devices.

To build such an attack, we leverage an open source ASR system Kaldi [21], which includes acoustic model and language model. By carefully synthesizing the outputs of the acoustic model from both the song and the given voice command, we are able to generate the adversarial audio with minimum perturbations through gradient descent, so that the CommanderSong can be less noticeable to human users (addressing C2, named WTA attack). To make such adversarial samples practical, our approach has been designed to capture the electronic noise produced by different speakers, and integrate a generic noise model into the algorithm for seeking adversarial samples (addressing C1, called WAA attack).
In our experiment, we generated over 200 CommanderSongs that contain different commands, and attacked Kaldi with an 100% success rate in a WTA attack and a 96% success rate in a WAA attack. Our evaluation further demonstrates that such a CommanderSong can be used to perform a black box attack on a mainstream ASR system iFLYTEK\(^1\) [19] (neither source code nor model is available). iFLYTEK has been used as the voice input method by many popular commercial apps, including WeChat (a social app with 963 million users), Sina Weibo (another social app with 530 million users), JD (an online shopping app with 270 million users), etc. To demonstrate the impact of our attack, we show that CommanderSong can be spread through YouTube, which might impact millions of users. To understand the human perception of the attack, we conducted a user study\(^2\) on Amazon Mechanical Turk [3]. Among over 200 participants, none of them identified the commands inside our CommanderSongs. Next, we will introduce overview of this attack, including motivations and philosophy of designing this attack.

3.2 Overview

In this section, we present the motivation of our work, and overview the proposed approach to generate the practical adversarial attack.

\(^1\)We have reported this to iFLYTEK, and are waiting for their responses.

\(^2\)The study is approved by the IRB.
3.2.1 Motivation

Recently, adversarial attacks on image classification have been extensively studied [36, 50]. Results show that even the state-of-the-art DNN-based classifier can be fooled by small perturbations added to the original image [70], producing erroneous classification results. However, the impact of adversarial attacks on the most advanced speech recognition systems, such as those integrating DNN models, has never been systematically studied. Hence, in this paper, we investigated DNN-based speech recognition systems, and explored adversarial attacks against them. Researches show that commands can be transmitted to IVC devices through inaudible ultrasonic sound [106] and noises [37]. Even though the existing works against ASR systems are seminal, they are limited in some aspects. Specifically, ultrasonic sound can be defeated by using a low-pass filter (LPF) or analyzing the signal frequency range, and noises are easy to be noticed by users.

Therefore, the research in this paper is motivated by the following questions: (Q1) Is it possible to build the practical adversarial attack against ASR systems, given the facts that the most ASR systems are becoming more intelligent (e.g., by integrating DNN models) and that the generated adversarial samples should work in the very complicated physical environment, e.g., electronic noise from speaker, background noise, etc.? (Q2) Is it feasible to generate the adversarial samples (including the target commands) that are difficult, or even impossible, to be noticed by ordinary users, so the
control over the ASR systems can happen in a “hidden” fashion? (Q3) If such adver-
sarial audio samples can be produced, is it possible to impact a large amount of victims
in an automated way, rather than solely relying on attackers to play the adversarial au-
dio and affecting victims nearby? Below, we will detail how our attack is designed to
address the above questions.

3.2.2 The Philosophy of Designing This Attack

To address Q3, our idea is to choose songs as the “carrier” of the voice commands
recognizable by ASR systems. The reason of choosing such “carrier” is at least two-
fold. On one hand, enjoying songs is always a preferred way for people to relax, e.g.,
listening to the music station, streaming music from online libraries, or just browsing
YouTube for favorite programs. Moreover, such entertainment is not restricted by us-
ing radio, CD player, or desktop computer any more. A mobile device, e.g., Android
phone or Apple iPhone, allows people to enjoy songs everywhere. Hence, choosing
the song as the “carrier” of the voice command automatically helps impact millions of
people. On the other hand, “hiding” the desired command in the song also makes the
command much more difficult to be noticed by victims, as long as Q2 can be reason-
ably addressed. Note that we do not rely on the lyrics in the song to help integrate the
desired command. Instead, we intend to avoid the songs with the lyrics similar to our
desired command. For instance, if the desired command is “open the door”, choosing a
song with the lyrics of “open the door” will easily catch the victims’ attention. Hence, we decide to use random songs as the “carrier” regardless of the desired commands.

Actually choosing the songs as the “carrier” of desired commands makes Q2 even more challenging. Our basic idea is when generating the adversarial samples, we revise the original song leveraging the pure voice audio of the desired command as a reference. In particular, we find the revision of the original song to generate the adversarial samples is always a trade off between preserving the fidelity of the original song and recognizing the desired commands from the generated sample by ASR systems. To better obfuscate the desired commands in the song, in this paper we emphasize the former than the latter. In other words, we designed our revision algorithm to maximally preserve the fidelity of the original song, at the expense of losing a bit success rate of recognition of the desired commands. However, such expense can be compensated by integrating the same desired command multiple times into one song (the command of “open the door” may only last for 2 seconds.), and the successful recognition of one suffices to impact the victims.

Technically, in order to address Q2, we need to investigate the details of an ASR system. As shown in Chapter 2, an ASR system is usually composed of two pre-trained models: an acoustic model describing the relationship between audio signals and phonetic units, and a language model representing statistical distributions over sequences of words. In particular, given a piece of pure voice audio of the desired command and a “carrier” song, we can feed them into an ASR system separately, and intercept the
intermediate results. By investigating the output from the acoustic model when processing the audio of the desired command, and the details of the language model, we can conclude the “information” in the output that is necessary for the language model to produce the correct text of the desired command. When we design our approach, we want to ensure such “information” is only a small subset (hopefully the minimum subset) of the output from the acoustic model. Then, we carefully craft the output from the acoustic model when processing the original song, to make it “include” such “information” as well. Finally, we inverse the acoustic model and the feature extraction together, to directly produce the adversarial sample based on the crafted output (with the “information” necessary for the language model to produce the correct text of the desired command).

Theoretically, the adversarial samples generated above can be recognized by the ASR systems as the desired command if directly fed as input to such systems. Since such input usually is in the form of a wave file (in “WAV” format) and the ASR systems need to expose APIs to accept the input, we define such attack as the WAV-To-API (WTA) attack. However, to implement a practical attack as in Q1, the adversarial sample should be played by a speaker to interact with IVC devices over the air. In this paper, we define such practical attack as WAV-Air-API (WAA) attack. The challenge of the WAA attack is when playing the adversarial samples by a speaker, the electronic noise produced by the loudspeakers and the background noise in the open air have significant impact on the recognition of the desired commands from the adversarial
Fig. 3.1 Result of decoding “Echo”.

samples. To address this challenge, we improve our approach by integrating a generic noise model to the above algorithm.

3.3 Approach

We implement our attack by addressing two technical challenges: (1) minimizing the perturbations to the song, so the distortion between the original song and the generated adversarial sample can be as unnoticeable as possible, and (2) making the attack practical, which means CommanderSong should be played over the air to compromise IVC devices. To address the first challenge, we proposed pdf-id sequence matching to incur minimum revision at the output of the acoustic model, and use gradient descent to generate the corresponding adversarial samples as in Section 3.3.2. The second challenge is addressed by introducing a generic noise model to simulate both the electronic noise and background noise as in Section 3.3.3. Below we elaborate the details.
### Table 3.1 Relationship between transition-id and pdf-id.

<table>
<thead>
<tr>
<th>Phoneme</th>
<th>HMM-state</th>
<th>Pdf-id</th>
<th>Transition-id</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$eh_B$</td>
<td>0</td>
<td>6383</td>
<td>15985</td>
<td>0→1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15986</td>
<td>0→2</td>
</tr>
<tr>
<td>$eh_B$</td>
<td>1</td>
<td>5760</td>
<td>16189</td>
<td>self-loop</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16190</td>
<td>1→2</td>
</tr>
<tr>
<td>$k_I$</td>
<td>0</td>
<td>6673</td>
<td>31223</td>
<td>0→1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>31224</td>
<td>0→2</td>
</tr>
<tr>
<td>$k_I$</td>
<td>1</td>
<td>3787</td>
<td>31379</td>
<td>self-loop</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>31380</td>
<td>1→2</td>
</tr>
<tr>
<td>$ow_E$</td>
<td>0</td>
<td>5316</td>
<td>39643</td>
<td>0→1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9644</td>
<td>0→2</td>
</tr>
<tr>
<td>$ow_E$</td>
<td>1</td>
<td>8335</td>
<td>39897</td>
<td>self-loop</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>39898</td>
<td>1→2</td>
</tr>
</tbody>
</table>

### 3.3.1 Kaldi Speech Recognition Platform

We choose the open source speech recognition toolkit Kaldi [21], due to its popularity in research community. Its source code on github obtains 3,748 stars and 1,822 forks [6]. Furthermore, the corpus trained by Kaldi on “Fisher” is also used by IBM [31] and Microsoft [103].

In order to use Kaldi to decode audio, we need a trained model to begin with. There are some models on Kaldi website that can be used for research. We took advantage of the “ASpIRE Chain Model” (referred as “ASpIRE model” in short), which was one of the latest released decoding models when we began our study\(^3\). After manually

\(^3\)There are three decoding models on Kaldi platform currently. ASpIRE Chain Model we used in this paper was released on October 15th, 2016, while SRE16 Xvector Model was released on October 4th, 2017, which was not available when we began our study. The CVTE Mandarin Model, released on June 21st 2017 was trained in Chinese [21].
analyzing the source code of Kaldi (about 301,636 lines of shell scripts and 238,107 C++ SLOC), we completely explored how Kaldi processes audio and decodes it to texts. Firstly, it extracts acoustic features like MFCC or PLP from the raw audio. Then based on the trained probability density function (p.d.f.) of the acoustic model, those features are taken as input to DNN to compute the posterior probability matrix. The p.d.f. is indexed by the pdf identifier (pdf-id), which exactly indicates the column of the output matrix of DNN.

Phoneme is the smallest unit composing a word. There are three states (each is denoted as an HMM state) of sound production for each phoneme, and a series of transitions among those states can identify a phoneme. A transition identifier (transition-id) is used to uniquely identify the HMM state transition. Therefore, a sequence of transition-ids can identify a phoneme, so we name such a sequence as phoneme identifier in this paper. Note that the transition-id is also mapped to pdf-id. Actually, during the procedure of Kaldi decoding, the phoneme identifiers can be obtained. By referring to the pre-obtained mapping between transition-id and pdf-id, any phoneme identifier can also be expressed as a specific sequence of pdf-ids. Such a specific sequence of pdf-ids actually is a segment from the posterior probability matrix computed from DNN. This implies that to make Kaldi decode any specific phoneme, we need to have DNN compute a posterior probability matrix containing the corresponding sequence of pdf-ids.
To illustrate the above findings, we use Kaldi to process a piece of audio with several known words, and obtain the intermediate results, including the posterior probability matrix computed by DNN, the transition-ids sequence, the phonemes, and the decoded words. Figure 2 demonstrates the decoded result of *Echo*, which contains three phonemes. The red boxes highlight the id representing the corresponding phoneme, and each phoneme is identified by a sequence of transition-ids, or the *phoneme identifiers*. Table 3.1 is a segment from the relationship among the phoneme, pdf-id, transition-id, etc. By referring to Table 3.1, we can obtain the pdf-ids sequence corresponding to the decoded transition-ids sequence\(^4\). Hence, for any posterior probability matrix demonstrating such a pdf-ids sequence should be decoded by Kaldi as *eh\(_B\).*

\(^4\)For instance, the pdf-ids sequence for *eh\(_B\)* should be 6383, 5760, 5760, 5760, 5760, 5760, 5760, 5760, 5760, 5760.
3.3.2 Gradient Descent to Craft Audio

Figure 3.2 demonstrates the details of our attack approach. Given the original song $x(t)$ and the pure voice audio of the desired command $y(t)$, we use Kaldi to decode them separately. By analyzing the decoding procedures, we can get the output of DNN matrix $A$ of the original song (Step 1 in Figure 3.2) and the phoneme identifiers of the desired command audio (Step 4 in Figure 3.2).

The DNN’s output $A$ is a matrix containing the probability of each pdf-id at each frame. Suppose there are $n$ frames and $k$ pdf-ids, let $a_{i,j}$ ($1 \leq i \leq n, 1 \leq j \leq k$) be the element at the $i$th row and $j$th column in $A$. Then $a_{i,j}$ represents the probability of the $j$th pdf-id at frame $i$. For each frame, we calculate the most likely pdf-id as the one with the highest probability in that frame. That is,

$$m_i = \arg \max_j a_{i,j}.$$  

Let $m = (m_1, m_2, \ldots, m_n)$. $m$ represents a sequence of most likely pdf-ids of the original song audio $x(t)$. For simplification, we use $g$ to represent the function that takes the original audio as input and outputs a sequence of most likely pdf-ids based on DNN’s predictions. That is,

$$g(x(t)) = m.$$
As shown in Step 5 in Figure 3.2, we can extract a sequence of pdf-id of the command $b = (b_1, b_2, \ldots, b_n)$, where $b_i$ ($1 \leq i \leq n$) represents the highest probability pdf-id of the command at frame $i$. To have the original song decoded as the desired command, we need to identify the minimum modification $\delta(t)$ on $x(t)$ so that $m$ is same or close to $b$. Specifically, we minimize the $L1$ distance between $m$ and $b$. As $m$ and $b$ are related with the pdf-id sequence, we define this method as pdf-id sequence matching algorithm.

Based on these observations we construct the following objective function:

$$\arg\min_{\delta(t)} \| g(x(t) + \delta(t)) - b \|_1.$$  \hspace{1cm} (3.1)

To ensure that the modified audio does not deviate too much from the original one, we optimize the objective function Eq (3.1) under the constraint of $|\delta(t)| \leq l$.

Finally, we use gradient descent [82], an iterative optimization algorithm to find the local minimum of a function, to solve the objective function. Given an initial point, gradient descent follows the direction which reduces the value of the function most quickly. By repeating this process until the value starts to remain stable, the algorithm is able to find a local minimum value. In particular, based on our objective function, we revise the song $x(t)$ into $x'(t) = x(t) + \delta(t)$ with the aim of making most likely pdf-ids $g(x'(t))$ equal or close to $b$. Therefore, the crafted audio $x'(t)$ can be decoded as the desired command.
To further preserve the fidelity of the original song, one method is to minimize the time duration of the revision. Typically, once the pure command voice audio is generated by a text-to-speech engine, all the phonemes are determined, so as to the phoneme identifiers and b. However, the speed of the speech also determines the number of frames and the number of transition-ids in a phoneme identifier. Intuitively, slow speech always produces repeated frames or transition-ids in a phoneme. Typically people need six or more frames to realize a phoneme, but most speech recognition systems only need three to four frames to interpret a phoneme. Hence, to introduce the minimal revision to the original song, we can analyze b, reduce the number of repeated frames in each phoneme, and obtain a shorter \( b' = (b_1, b_2, \ldots, b_q) \), where \( q < n \).

### 3.3.3 Practical Attack Over The Air

By feeding the generated adversarial sample directly into Kaldi, the desired command can be decoded correctly. However, playing the sample through a speaker to physically attack an IVC device typically cannot work. This is mainly due to the noises introduced by the speaker and environment, as well as the distortion caused by the receiver of the IVC device. In this paper, we do not consider the invariance of background noise in different environments, e.g., grocery, restaurant, office, etc., due to the following reasons: (1) In a quite noisy environment like restaurant or grocery, even the original voice command \( y(t) \) may not be correctly recognized by IVC devices; (2) Modeling any slightly variant background noise itself is still an open research prob-
lem; (3) Based on our observation, in a normal environment like home, office, lobby, the major impacts on the physical attack are the electronic noise from the speaker and the distortion from the receiver of the IVC devices, rather than the background noise.

Hence, our idea is to build a noise model, considering the speaker noise, the receiver distortion, as well as the generic background noise, and integrate it in the approach in Section 3.3.2. Specifically, we carefully picked up several songs and played them through our speaker in a very quiet room. By comparing the recorded audio (captured by our receiver) with the original one, we can capture the noises. Note that playing “silent” audio does not work since the electronic noise from speakers may depend on the sound at different frequencies. Therefore, we intend to choose the songs that cover more frequencies. Regarding the comparison between two pieces of audio, we have to first manually align them and then compute the difference. We redesign the objective function as shown in Eq (3.2).

$$\arg\min_{\mu(t)} \|g(x(t) + \mu(t) + n(t)) - b\|_1,$$  \hspace{1cm} (3.2)

where $\mu(t)$ is the perturbation that we add to the original song, and $n(t)$ is the noise samples that we captured. In this way, we can get the adversarial audio $x'(t) = x(t) + \mu(t)$ that can be used to launch the practical attack over the air.

Such noise model above is quite device-dependent. Since different speakers and receivers may introduce different noises/distortion when playing or receiving specific
audio, $x'(t)$ may only work with the devices that we use to capture the noise. To enhance the robustness of $x'(t)$, we introduce random noise, which is shown in Eq (3.3). Here, the function $\text{rand}(\cdot)$ returns an vector of random numbers in the interval (-N,N), which is saved as a “WAV” format file to represent $n(t)$. Our evaluation results show that this approach can make the adversarial audio $x'(t)$ robust enough for different speakers and receivers.

$$n(t) = \text{rand}(t), |n(t)| \leq N. \quad (3.3)$$

### 3.4 Evaluation

In this section, we present the experimental results of CommanderSong. We evaluated both the WTA and WAA attacks against machine recognition. To evaluate the human comprehension, we conducted a survey examining the effects of “hiding” the desired command in the song. Then, we tested the transferability of the adversarial sample on other ASR platforms, and checked whether CommanderSong can spread through Internet and radio. Finally, we measured the efficiency in terms of the time to generate the CommanderSong. Demos of attacks are uploaded on the website https://sites.google.com/view/commandersong/.
<table>
<thead>
<tr>
<th>Command</th>
<th>Success rate (%)</th>
<th>SNR (dB)</th>
<th>Efficiency (frames/hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Okay google restart phone now.</td>
<td>100</td>
<td>18.6</td>
<td>229/1.3</td>
</tr>
<tr>
<td>Okay google flashlight on.</td>
<td>100</td>
<td>14.7</td>
<td>219/1.3</td>
</tr>
<tr>
<td>Okay google read mail.</td>
<td>100</td>
<td>15.5</td>
<td>217/1.5</td>
</tr>
<tr>
<td>Okay google clear notification.</td>
<td>100</td>
<td>14</td>
<td>260/1.2</td>
</tr>
<tr>
<td>Okay google good night.</td>
<td>100</td>
<td>15.6</td>
<td>193/1.3</td>
</tr>
<tr>
<td>Okay google airplane mode on.</td>
<td>100</td>
<td>16.9</td>
<td>219/1.1</td>
</tr>
<tr>
<td>Okay google turn on wireless hot spot.</td>
<td>100</td>
<td>14.7</td>
<td>280/1.6</td>
</tr>
<tr>
<td>Okay google read last sms from boss.</td>
<td>100</td>
<td>15.1</td>
<td>323/1.4</td>
</tr>
<tr>
<td>Echo open the front door.</td>
<td>100</td>
<td>17.2</td>
<td>193/1.0</td>
</tr>
<tr>
<td>Echo turn off the light.</td>
<td>100</td>
<td>17.3</td>
<td>347/1.5</td>
</tr>
<tr>
<td>Okay google call one one zero one nine one two zero.</td>
<td>100</td>
<td>14.8</td>
<td>387/1.7</td>
</tr>
<tr>
<td>Echo ask capital one to make a credit card payment.</td>
<td>100</td>
<td>15.8</td>
<td>379/1.9</td>
</tr>
</tbody>
</table>

### 3.4.1 Experiment Setup

The pure voice audio of the desired commands can be generated by any Text-To-Speech (TTS) engine (e.g., Google text-to-speech [13], etc.) or recording human voice, as long as it can be correctly recognized by Kaldi platform. We also randomly downloaded 26 songs from the Internet. To understand the impact of using different types of songs as the carrier, we intended to choose songs from different categories, i.e., popular, rock, rap, and soft music. Regarding the commands to inject, we chose 12 commonly used ones such as “turn on GPS”, “ask Capital One to make a credit card...
payment”, etc., as shown in Table 3.2. Regarding the computing environment, one GPU server (1075MHz GPU with 12GB memory, and 512GB hard drive) was used.

### 3.4.2 Effectiveness

In this subsection, the details of WTA and WAA attack effectiveness will be discussed.

**WTA Attack.** In this WTA attack, we directly feed the generated adversarial songs to Kaldi using its exposed APIs, which accept raw audio file as input. Particularly, we injected each command into each of the downloaded 26 songs using the approach proposed in Section 3.3.2. Totally we got more than 200 adversarial songs in the “WAV” format and sent them to Kaldi directly for recognition. If Kaldi successfully identified the command injected inside, we denote the attack as successful.

Table 3.2 shows the WTA attack results. Each command can be recognized by Kaldi correctly. The success rate 100% means Kaldi can decode every word in the desired command correctly. The success rate is calculated as the ratio of the number of words successfully decoded and the number of words in the desired command. Note in the case that the decoded word is only one character different than that in the desired command, we consider the word is not correctly recognized.

For each adversarial song, we further calculated the average signal-noise ratio (SNR) against the original song as shown in Table 3.2. SNR is a parameter widely used to quantify the level of a signal power to noise, so we use it here to measure the distortion of the adversarial sample over the original song. We then use the following
Table 3.3 WAA attack results.

<table>
<thead>
<tr>
<th>Command</th>
<th>Speaker</th>
<th>Success rate (%)</th>
<th>SNR (dB)</th>
<th>Efficiency (frames/hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Echo ask capital one to make a credit card payment.</td>
<td>JBL speaker</td>
<td>90</td>
<td>1.7</td>
<td>379/2.0</td>
</tr>
<tr>
<td></td>
<td>ASUS Laptop</td>
<td>82</td>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SENMATE Broadcast</td>
<td>72</td>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>Okay google call one zero one one nine one two zero.</td>
<td>JBL speaker</td>
<td>96</td>
<td>1.3</td>
<td>400/1.8</td>
</tr>
<tr>
<td></td>
<td>ASUS Laptop</td>
<td>60</td>
<td>1.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SENMATE Broadcast</td>
<td>70</td>
<td>1.3</td>
<td></td>
</tr>
</tbody>
</table>

The equation $SNR(dB) = 10 \log_{10}(P_x(t)/P_\delta(t))$ is used to obtain SNR, where the original song $x(t)$ is the signal while the perturbation $\delta(t)$ is the noise. Larger SNR value indicates a smaller perturbation. Based on the results in Table 3.2, the SNR ranges from 14~18.6 $dB$, indicating that the perturbation in the original song is less than 4%. Therefore, the perturbation should be too slight to be noticed.

**WAA Attack.** To practically attack Kaldi over the air, the ideal case is to find a commercial IVC device implemented based on Kaldi and play our adversarial samples against the device. However, we are not aware of any such IVC device, so we simulate a pseudo IVC device based on Kaldi. In particular, the adversarial samples are played by speakers over the air. We use the recording functionality of iPhone 6S to record the audio, which is sent to Kaldi API to decode. Overall, such a pseudo IVC device is built using the microphone in iPhone 6S as the audio recorder, and Kaldi system to decode the audio.

We conducted the practical WAA attack in a meeting room (16 meter long, 8 meter wide, and 4 meter tall). The songs were played using three different speakers.
including a JBL clip2 portable speaker, an ASUS laptop and a SENMATE broadcast equipment [26], to examine the effectiveness of the injected random noise. All of the speakers are easy to obtain and carry. The distance between the speaker and the pseudo IVC device (i.e., the microphone of the iPhone 6S) was set at 1.5 meters. We chose two commands as in Table 3.3, and generated adversarial samples. Then we played them over the air using those three different speakers and used the iPhone 6S to record the audios, which were sent to Kaldi to decode. Table 3.3 shows the WAA attack results. For both of the two commands, JBL speaker overwhelms the other two with the success rate up to 96%, which might indicate its sound quality is better than the other two. All the SNRs are below 2 dB, which indicates slightly bigger perturbation to the original songs due to the random noise from the signal’s point of view. Below we will evaluate if such “bigger” perturbation is human-noticeable by conducting a survey.

**Human comprehension from the survey.** To evaluate the effectiveness of hiding the desired command in the song, we conducted a survey on Amazon Mechanical Turk (MTurk) [3], an online marketplace for crowdsourcing intelligence. We recruited 204 individuals to participate in our survey⁵. Each participant was asked to listen to 26 adversarial samples, each lasting for about 20 seconds (only about four or five seconds

---

⁵The survey will not cause any potential risks to the participants (physical, psychological, social, legal, etc.). The questions in our survey do not involve any confidential information about the participants. We obtained the IRB Exempt certificates from our institutes in https://github.com/RiskySignal/Devil-Whisper-Attack.
in the middle is crafted to contain the desired command.). A series of questions regarding each audio need to be answered, e.g., (1) whether they have heard the original song before; (2) whether they heard anything abnormal than a regular song (The four options are no, not sure, noisy, and words different than lyrics); (3) if choosing noisy option in (2), where they believe the noise comes from, while if choosing words different than lyrics option in (2), they are asked to write down those words, and how many times they listened to the song before they can recognize the words.

Table 3.4 Human comprehension of the WTA samples.

<table>
<thead>
<tr>
<th>Music Classification</th>
<th>Listened (%)</th>
<th>Abnormal (%)</th>
<th>Recognize Command (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soft Music</td>
<td>13</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>Rock</td>
<td>33</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>Popular</td>
<td>32</td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>Rap</td>
<td>41</td>
<td>23</td>
<td>0</td>
</tr>
</tbody>
</table>

The entire survey lasts for about five to six minutes. Each participant is compensated $0.3 for successfully completing the study, provided they pass the attention check question to motivate the participants concentrate on the study. Based on our study, 63.7% of the participants are in the age of 20~40 and 33.3% are 40~60 years old, and 70.6% of them use IVC devices (e.g., Amazon Echo, Google home, Smartphone, etc.) everyday.

Table 3.4 shows the results of the human comprehension of our WTA samples. We show the average results for songs belonging to the same category. Generally, the songs in soft music category are the best candidates for the carrier of the desired command,
Table 3.5 Human comprehension of the WAA samples.

<table>
<thead>
<tr>
<th>Song Name</th>
<th>Listened (%)</th>
<th>Abnormal (%)</th>
<th>Noise-speaker (%)</th>
<th>Noise-song (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Did You Need It</td>
<td>15</td>
<td>67</td>
<td>42</td>
<td>1</td>
</tr>
<tr>
<td>Outlaw of Love</td>
<td>11</td>
<td>63</td>
<td>36</td>
<td>2</td>
</tr>
<tr>
<td>The Saltwater Room</td>
<td>27</td>
<td>67</td>
<td>39</td>
<td>3</td>
</tr>
<tr>
<td>Sleepwalker</td>
<td>13</td>
<td>67</td>
<td>41</td>
<td>0</td>
</tr>
<tr>
<td>Underneath</td>
<td>13</td>
<td>68</td>
<td>45</td>
<td>3</td>
</tr>
<tr>
<td>Feeling Good</td>
<td>38</td>
<td>59</td>
<td>36</td>
<td>4</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>19.5</strong></td>
<td><strong>65.2</strong></td>
<td><strong>40</strong></td>
<td><strong>2.2</strong></td>
</tr>
</tbody>
</table>

with as low as 15% of participants noticed the abnormality. None of the participants could recognize any word of the desired command injected in the adversarial samples of any category. Table 3.5 demonstrates the results of the human comprehension of our WAA samples. On average, 40% of the participants believed the noise was generated by the speaker or like radio, while only 2.2% of them thought the noise from the samples themselves. In addition, less than 1% believed that there were other words except the original lyrics. However, none of them successfully identified any word even repeating the songs several times.

3.4.3 Towards the Transferability

Finally, we assess whether the proposed CommanderSong can be transferred to other ASR platforms.

**Transfer from Kaldi to iFLYTEK.** We choose iFLYTEK ASR system as the target of our transfer, due to its popularity. As one of the top five ASR systems in the world, it possesses 70% of the market in China. In particular, *iFLYTEK Input* is a
popular mobile voice input method, which supports mandarin, English and personalized input [20]. *iFLYREC* is an online service offered by iFLYTEK to convert audio to text [18]. We use them to test the transferability of our WAA attack samples, and the success rates of different commands are shown in Table 3.6. Note that WAA audio samples are directly fed to *iFLYREC* to decode. Meanwhile, they are played using Bose Companion 2 speaker towards *iFLYTEK Input* running on smartphone LG V20, or using JBL speaker towards *iFLYTEK Input* running on smartphone Huawei honor 8/MI note3/iPhone 6S. Those adversarial samples containing commands like *open the door* or *good night* can achieve great transferability on both platforms. However, the command *airplane mode on* only gets 66% success rate on *iFLYREC*, and 0 on *iFLYTEK Input*.

**Transferability from Kaldi to DeepSpeech.** We also try to transfer Commander-Song from Kaldi to DeepSpeech, which is an open source end-to-end ASR system. We directly fed several adversarial WTA and WAA attack samples to DeepSpeech, but none of them can be decoded correctly. As Carlini et al. have successfully modified any audio into a command recognizable by DeepSpeech [39], we intend to leverage their open source algorithm to examine if it is possible to generate one adversarial sample against both two platforms. In this experiment, we started by 10 adversarial samples generated by CommanderSong, either WTA or WAA attack, integrating commands like *Okay google call one one zero one one nine one two zero, Echo open the*
Table 3.6 Transferability from Kaldi to iFLYTEK.

<table>
<thead>
<tr>
<th>Command</th>
<th>iFLYREC (%)</th>
<th>iFLYTEK Input (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airplane mode on.</td>
<td>66</td>
<td>0</td>
</tr>
<tr>
<td>Open the door.</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Good night.</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

front door, and Echo turn off the light. We applied their algorithm to modify the samples until DeepSpeech can decode the target commands correctly. Then we tested such newly generated samples against Kaldi as WTA attack, and Kaldi can still successfully recognize them. We did not perform WAA attack since their algorithm targeting DeepSpeech cannot achieve attacks over the air.

The preliminary evaluations on transferability give us the opportunities to understand CommanderSongs and for designing systematic approach to transfer in the future.

3.4.4 Automated Spreading

Since our WAA attack samples can be used to launch the practical adversarial attack against ASR systems, we want to explore the potential channels that can be leveraged to impact a large amount of victims automatically.

**Online sharing.** We consider the online sharing platforms like YouTube to spread CommanderSong. We picked up one five-second adversarial sample embedded with the command “open the door” and applied Windows Movie Maker software to make
a video, since YouTube only supports video uploading. The sample was repeated four times to make the full video around 20 seconds. We then connected our desktop audio output to Bose Companion 2 speaker and installed *iFLYTEK Input* on LG V20 smartphone. In this experiment, the distance between the speaker and the phone can be up to 0.5 meter, and *iFLYTEK Input* can still decode the command successfully.

**Radio broadcasting.** In this experiment, we used HackRF One [14], a hardware that supports Software Defined Radio (SDR) to broadcast our CommanderSong at the frequency of FM 103.4 MHz, simulating a radio station. We setup a radio at the corresponding frequency, so it can receive and play the CommanderSong. We ran the *WeChat* application and enabled the *iFLYTEK Input* on different smartphones including iPhone 6S, Huawei Honor 8 and XiaoMi MI Note3. *iFLYTEK Input* can always successfully recognize the command “open the door” from the audio played by the radio and display it on the screen.

### 3.4.5 Efficiency

We also evaluate the cost of generating CommanderSong in the aspect of the required time. For each command, we record the time to inject it into different songs and compute the average. Since the time required to craft also depends on the length of the desired command, we define the efficiency as the ratio of the number of frames

---

*6WeChat* is the most popular instant messaging application in China, with approximately 963,000,000 users all over the world by June 2017.
Fig. 3.3 SNR impacts on correlation of the audios and the success rate of adversarial audios.

of the desired command and the required time. Table 3.2 and Table 3.3 show the efficiency of generating WTA and WAA samples for different commands. Most of those adversarial samples can be generated in less than two hours, and some simple commands like “Echo open the front door” can be done within half an hour. However, we do notice that some special words (such as GPS and airplane) in the command make the generation time longer. Probably those words are not commonly used in the training process of the “ASpIRE model” of Kaldi, so generating enough phonemes to represent the words is time-consuming. Furthermore, we find that, for some songs in the rock category such as “Bang bang” and “Roaked”, it usually takes longer to generate the adversarial samples for the same command compared with the songs in other categories, probably due to the unstable rhythm of them.
3.5 Understanding the Attacks

We try to deeply understand the attacks, which could potentially help to derive defense approaches. We raise some questions and perform further analysis on the attacks.

**In what ways does the song help the attack?** We use songs as the carrier of commands to attack ASR systems. Obviously, one benefit of using a song is to prevent listeners from being aware of the attack. Also CommanderSong can be easily spread through Youtube, radio, TV, etc. Does the song itself help generate the adversarial audio samples? To answer this question, we use a piece of silent audio as the “carrier” to generate CommanderSong $A_{cs}$ (WAA attack), and test the effectiveness of it. The results show that $A_{cs}$ can work, which is aligned with our findings - a random song can serve as the “carrier” because a piece of silent audio can be viewed as a special song.

However, after listening to $A_{cs}$, we find that $A_{cs}$ sounds quite similar to the injected command, which means any user can easily notice it, so $A_{cs}$ is not the adversarial samples we desire. Note that, in our human subject study, none of the participants recognized any command from the generated CommanderSongs. We assume that *some phonemes or even smaller units in the original song work together with the injected small perturbations to form the target command*. To verify this assumption, we prepare a song $A_s$ and use it to generate the CommanderSong $A_{cs}$. Then we calculate the difference $\Delta(A_s, A_{cs})$ between them, and try to attack ASR systems using $\Delta(A_s, A_{cs})$. However, after several times of testing, we find that $\Delta(A_s, A_{cs})$ does not work, which
Fig. 3.4 Explanation of Kaldi and human recognition of the audios.

indicates the pure perturbations we injected cannot be recognized as the target commands.

Recall that in Table 3.5, the songs in the soft music category are proven to be the best carrier, with lowest abnormality identified by participants. Based on the findings above, it appears that such songs can better aligned with the phonemes or smaller “units” in the target command to help the attack. This is also the reason why \( \Delta(A_s, A_{cs}) \) cannot directly attack successfully: the “units” in the song combined with \( \Delta(A_s, A_{cs}) \) together construct the phonemes of the target command.

What is the impact of noise in generating adversarial samples? As mentioned early, we build a generic random noise model to perform the WAA attack over the air. In order to understand the impact of the noise in generating adversarial samples, we crafted CommanderSong using noises with different amplitude values. Then we observed the differences between the CommanderSong and the original song, the
differences between the CommanderSong and the pure command audio, and the success rates of the CommanderSong to attack. To characterize the difference, we leverage Spearman’s rank correlation coefficient [85] (Spearman’s rho for short) to represent the similarity between two pieces of audio. Spearman’s rho is widely used to represent the correlation between two variables, and can be calculated as follows:

\[ r(X, Y) = \frac{Cov(X, Y)}{\sqrt{Var[X]Var[Y]}}, \]

where \( X \) and \( Y \) are the MFCC features of the two pieces of audio. \( Cov(X, Y) \) represents the covariance of \( X \) and \( Y \). \( Var[X] \) and \( Var[Y] \) are the variances of \( X \) and \( Y \) respectively.

The results are shown in Figure 3.3. The x-axis in the figure shows the \( SNR \) (in \( dB \)) of the noise, and the y-axis gives the correlation. From the figure, we find that the correlation between the CommanderSong and the original song (red line) decreases with \( SNR \). It means that the CommanderSong sounds less like the original song when the amplitude value of the noise becomes larger. This is mainly because the original song has to be modified more to find a CommanderSong robust enough against the introduced noise. On the contrary, the CommanderSong becomes more similar with the target command audio when the amplitude values of the noise increases (i.e., decrease of \( SNR \) in the figure, blue line), which means that the CommanderSong sounds more like the target command. The success rate (black dotted line) also increases with the decrease of \( SNR \). We also note that, when \( SNR = 4 \) \( dB \), the success rate could be as high as 88%. Also the correlation between CommanderSong and the original song is 90%, which indicates a high similarity.
Figure 3.4 shows the results from another perspective. Suppose the dark blue circle is the set of audios that can be recognized as commands by ASR systems, while the light blue circle and the red one represent the sets of audio recognized as commands and songs by human respectively. At first, the original song is in the red circle, which means that neither ASR systems nor human being recognize any command inside. WTA attack slightly modifies the song so that the open source system Kaldi can recognize the command while human cannot. After noises are introduced to generate CommanderSong for WAA attacks, CommanderSong will fall into the light blue area step by step, and in the end be recognized by human. Therefore, attackers can choose the amplitude values of noise to balance between robustness to noise and identifiability by human users.

### 3.6 Defense

We propose two approaches to defend against CommanderSong: Audio turbulence and Audio squeezing. The first defense is effective against WTA, but not WAA; while the second defense works against both attacks.
**Audio turbulence.** From the evaluation, we observe that noise (e.g., from speaker or background) decreases the success rate of CommanderSong while impacts little on the recognition of audio command. So our basic idea is to add noise (referred to as turbulence noise $A_n$) to the input audio $A_I$ before it is received by the ASR system, and check whether the resultant audio $A_I \oplus A_n$ can be interpreted as other words. Particularly, as shown in Figure 3.5, $A_I$ is decoded as $text_1$ by the ASR system. Then we add $A_n$ to $A_I$ and let the ASR system extract the text $text_2$ from $A_I \oplus A_n$. If $text_1 \neq text_2$, we say that the CommanderSong is detected.

![Fig. 3.6 The results of audio turbulence defense.](image)

We did experiments using this approach to test the effectiveness of such defense. The target command “open the door” was used to generate a CommanderSong. Figure 3.6 shows the result. The x-axis shows the $SNR (A_I$ to $A_n$), and the y-axis shows the success rate. We found that the success rate of WTA dramatically drops when $SNR$ decreases. When $SNR = 15 \text{ dB}$, WTA almost always fails and $A_I$ can still be successfully recognized, which means this approach works for WTA. However, the
success rate of WAA is still very high. This is mainly because CommanderSongs for WAA is generated using random noises, which is robust against turbulence noise.

![Audio squeezing defense result.](image)

**Fig. 3.7** Audio squeezing defense result.

**Audio squeezing.** The second defense is to reduce the sampling rate of the input audio $A_I$ (just like squeezing the audio). Instead of adding $A_n$ in the defense of audio turbulence, we downsample $A_I$ (referred to as $D(A_I)$). Still, ASR systems decode $A_I$ and $D(A_I)$, and get $\text{text}_1$ and $\text{text}_2$ respectively. If $\text{text}_1 \neq \text{text}_2$, the CommanderSong is detected. Similar to the previous experiment, we evaluate the effectiveness of this approach. The results are shown in Figure 3.7. The x-axis shows the ratio (1/$M$) of downsampling ($M$ is the downsampling factor or decimation factor, which means that the original sampling rate is $M$ times of the downsampled rate). When $1/M = 0.7$ (if the sample rate is 8000 samples/second, the downsampled rate is 5600 samples/second), the success rates of WTA and WAA are 0% and 8% respectively. $A_I$ can still be successful recognized at the rate of 91%. This means that Audio squeezing is effective to defend against both WTA and WAA.
CHAPTER 4

Black Box Audio Adversarial Attack

In this chapter, we will present the details of Devil’s Whisper attack - a practical audio adversarial attack towards several commercial black box speech recognition devices like Amazon Echo.

4.1 Introduction

With the advance of automatic speech recognition (ASR) technologies, intelligent voice control (IVC) devices become increasingly popular. Today, smart speakers like Google Home, Amazon Echo, Apple HomePod are already part of our daily life. Also the availability of ASR services such as Google Cloud Speech-to-Text [12], Amazon Transcribe [4], Microsoft Bing Speech Service [23] and IBM Speech to Text [16] enable their users to conveniently integrate their APIs to control smart devices, conduct long-form audio transcription, text analysis, video analysis and etc. More recently, Amazon introduces Auto SDK that allows drivers to interact with vehicles using voice
commands. However, the extensive use of voice for critical system control also brings
in security concerns, whose implications have not yet been fully understood.

**AE threats to ASR.** More specifically, voice is an open channel and therefore the
commands received by IVC devices could come from any source. In recent years, re-
searchers have shown that unauthorized voice commands can be injected into wireless
signals [65], in the form of noise [37] or even inaudible ultrasound [106], to stealthily
gain control of the IVC devices. Recently, attempts have been made to utilize ad-
versarial examples (AEs), which are found to be effective against image processing
systems [95], to exploit ASR systems. Particularly, Carlini et al. [39] have success-
fully attacked DeepSpeech (the open-source ASR model of Mozilla) using AEs, with
the full knowledge of model parameters. Yuan et al. proposed CommanderSong [105]
that automatically generates AEs embedded into songs to attack open-source Kaldi
Aspire Chain Model [21] over-the-air.

These approaches demonstrate that the real-world ASR systems are vulnerable in
a *white-box model*, when their internal parameters are exposed to the adversary. Less
clear is the security risks the commercial ASR systems such as Google Home, Mi-
crosoft Cortana, Amazon Echo and Apple Siri are facing. Recently, Taori et al. have
made the targeted adversarial attack by treating DeepSpeech as a black-box [96]. How-
ever, so far no success has been reported when it comes to generating AEs against the
deep learning models behind commercial, close-source ASR systems, up to our knowl-
edge.
Black-box AE attacks on ASR systems are difficult. In addition to the challenge introduced by the lack of information about the target’s model and parameters, as also faced by the black-box attacks on image processing [84], an ASR system tends to be more complicated than an image recognition system, due to its complicated architecture, including feature extraction, acoustic model and language model, and the design for processing a time series of speech data. As evidenced in our study, when directly applying the existing technique to build a substitute on the data labeled by the target [84], we found that about 24 hours training set (require around 5100 oracle queries with each audio around 25 seconds), even with a target-based optimization (Section 4.3.2.1), only gives us a substitute model with merely 25% transferability against Google Cloud Speech-to-Text API command_and_search model (Section 4.6.4). By comparison, prior research reports that the similar attack on image recognition systems like Google, Amazon and MetaMind APIs using simple datasets like MNIST with 800 queries to achieve a transferability rate over 90% [84].

**Devil’s Whisper.** We demonstrate that a black-box attack on the commercial ASR system and even device is completely feasible. Our attack, called Devil’s Whisper, can automatically generate audio clips as AEs against commercial ASR systems like Google Cloud Speech-to-Text API. These “hidden” target commands are stealthy for human being but can be recognized by these systems, which can lead to control of commercial IVC devices like Google Home. Our key idea is to use a small number of strategic queries to build a substitute model and further enhance it with an open-source
ASR, which helps address the complexity in the target system. More specifically, to construct the substitute, we utilize Text-to-Speech (TTS) API to synthesize commands audio clips, then we enlarge the corpus by before sending them as queries to the target. This allows us to focus on the types of the data most important to the success of our attack and makes the substitute model more approximate to the target. The substitute model trained over the data is then used in an ensemble learning together with an open-source ASR model (called base model). The AEs cross-generated by both models are systematically selected to attack the target.

In our experiment, we build substitute models approximating each of the four black-box speech API services (Google Cloud Speech-to-Text, Microsoft Bing Speech Service, IBM Speech to Text and Amazon Transcribe). Just over 4.6-hour training data (about 1500 queries with each audio about 25 seconds) is needed to ensure successful conversion of nearly 100% target commands into workable AEs\(^1\) when attacking most of the API services. Our AEs can also attack the corresponding black-box IVC devices\(^2\) (Google Assistant, Google Home, Microsoft Cortana and Amazon Echo) over-the-air with 98% of target commands successful. Furthermore, our AEs can be suc-

---

\(^1\)In this paper, we consider an AE “workable” or “successful” if it can either 1) be decoded by the target API service (converted into text) as expected in an API attack, or 2) cause the target IVC device to execute the target commands at least twice when playing the AE against the device over-the-air for no more than 30 times. Note that an over-the-air attack on device can be sensitive to environmental factors like volume, distance, device etc., while the attack on APIs is usually stable.

\(^2\)We have contacted the vendors and are waiting for their responses.
cessfully transferred to other black-box platforms, which have no public API services (e.g., Apple Siri). The user study on Amazon Mechanical Turk shows that none of the participants can identify any command from our AEs if they listen to them once.

4.2 Overview

In this section, we present the motivations, threat models of our work, and overview the challenges to conduct this work.

4.2.1 Motivation

In the era of Internet of Things (IoT), the voice-enabled centralized control devices are becoming more and more popular, e.g., Google Home, Amazon Echo, etc. Various smart home devices, like smart lock, smart light, smart switch can be paired to such “hub”, which allows them to be controlled naturally via voice. Moreover, the voice-assistant applications on smartphones or tablets, e.g., Google Assistant, Apple Siri, etc., offer a convenient way for people to use their mobile devices. In this paper, we use IVC devices to refer to all the above mentioned voice-enabled centralized control devices and smartphones or tablets.

An example for the potential security risk to the IVC system is smartphone navigation, which is widely used today to help drive through unfamiliar areas. Previous work [104] shows that the FM radio channel can be controlled by attackers to broadcast their malicious signals. Therefore, if the attackers craft their AE hiding a hostile
navigation command and broadcast it on the selected FM radio channel, those who run smartphone navigation while listening to the corresponding FM channel will be impacted. Actually, our experimental results show that “Okay Google, navigate to my home” can stealthily command Google Assistant on smartphones through music and none of the participants in our user study were able to identify the hidden command even after listening to the AE twice. This attack, if successful, will put both drivers and passengers to serious danger. Given the pervasiveness of the commercial IVC systems, it is important to understand whether such an attack is indeed realistic, particularly when the adversary has little information about how such systems work. Our research, therefore, aims at finding the answer.

To hack the commercial IVC devices in the real world successfully, there are generally two requirements for the attacks: (R1) effectiveness (towards device) and (R2) concealing (towards human). Both of the two requirements emphasize the practical aspects of such attacks, that is, to deceive those devices successfully but uninterpretable by human. Unfortunately, most of existing adversarial attacks fail either (R1) [39] or (R2) [37] in some extents. Hence, we concentrate on the research question “whether it is possible to hack those commercial IVC devices (mostly black-box based) in the real world with both (R1) and (R2) satisfied” in this paper.
4.2.2 Threat Model

Since our target is the commercial IVC devices, they are black-box to us by default. Specifically, we have no knowledge of the internals of the speech recognition systems, e.g., model parameters or hyperparameters. Instead, we assume the corresponding online Speech-to-Text API services, i.e., providing real time decoding results from input audio, are open to public. This assumption is valid for most of the popular IVC devices available on the market, e.g., Google Cloud Speech-to-Text API, Microsoft Bing Speech Service API, etc\(^3\). Either free or pay as you go, such services are accessible to third party developers. We further assume that for the same platform, the ASR system used to provide online speech API service and that used for the IVC devices are the same or similar\(^4\), e.g., Microsoft Bing Speech Service API and Microsoft Cortana.

Once the attack audio is generated, we assume it will be played by speakers (either dedicated speakers or speakers on radio, TV, smartphone, computer, etc.), which is placed not quite far away (e.g., 5\(^-\)200 centimeters) from the target IVC devices. For example, the methods proposed in [104] can be used to remotely control the contents played by the radio. Furthermore, we do not have the knowledge of the speakers, or

---

\(^3\)However, as the paper is written, we could not find such API service from Apple yet. Communication with Apple Siri developers confirmed that Apple has not released their speech API service to the public. In this work, we proposed an alternative approach to hack such IVC devices without corresponding API service available, like Apple Siri, in Section 4.6.3.

\(^4\)Based on our experiments, Amazon seems like an exception, which will be discussed in Section 4.6.2.
the microphones of the target devices. Once the attack is successful, an indicator could be observed. For instance, the attack audio with the command of “Echo, turn off the light” is successful by observing the corresponding light off.

4.2.3 Technical Challenges

Currently there are several methods to attack black-box models. First, attackers can probe the black-box model by continuously querying it with different inputs, analyzing the corresponding outputs, and adjusting the inputs by adding perturbations to craft the workable AEs. However, such method normally suffers from the problems of uncertainty in terms of probing process and timing cost, especially for a commercial IVC device whose models are quite complex for approximation. Another method is “transferability” based, i.e., AEs generated on a known Model $A$ are used to attack the target Model $B$, as long as those two models are similar in the aspects of algorithm, training data and model structure. If Model $A$ is hard to find, a local model can be trained based on the algorithm and training data to approximate the target Model $B$, to implement the “transferability”. However, since the target Model $B$ is black-box, the similarity is hard to determine and the algorithm as well as the training data may not be available.
4.3 Approaches

In this section, we present our approach of AE based attacks against the commercial black-box IVC devices. Figure 4.1 gives the details of our approach. We start by transferability based approach (Step ① in Figure 4.1), via an enhancement over the existing state-of-the-art work generating AEs against ASR systems. Then we describe the novel approach of “Alternate Models based Generation” (Step ②, ③, and ④ in Figure 4.1).

4.3.1 Transferability Based Approach

For the black-box AE based attacks, the knowledge about the internal model is not known, so a straightforward method is to generate AEs based on a white-box model and transfer the AEs to the black-box model. The success of the transferability based attacks depends on the similarity between the internal structure and parameters of the white-box and black-box models. Recent research demonstrates that the transferability-
ity could work on heterogeneous models through the improvement of AE generation algorithm [84].

**Initial try.** To implement the transferability-based approach, we start by adopting Kaldi ASpIRE Chain Model as the white-box model, and refer to the idea of “pdf-id matching algorithm” proposed in CommanderSong [105] to generate AEs. We make such choices because (i) CommanderSong is the state-of-the-art AE generation work based on white-box model as this paper is written; (ii) the AEs generated in CommanderSong demonstrates transferability to iFLYTEK application—a black-box ASR system—running on smartphone, when played over-the-air; and (iii) the white-box model used in CommanderSong is accessible and popular.

We tested the AEs generated using the above approach on our target black-box ASR systems such as the Google Cloud Speech-to-text API, and find that only few AEs can be partially recognized as “Google”, “Hey Google”, “phone”, etc. The success rate of the transferability on Amazon Transcribe, the API service offered by Amazon, is even lower. This is not surprising, since CommanderSong was not designed to transfer across different systems.

**Enhancement.** We analyzed the approach proposed in CommanderSong and enhanced it by applying the Momentum based Iterative Fast Gradient Method (MI-FGM) to improve the transferability of the AEs. The momentum method was introduced in [48], which can accumulate a velocity vector in the gradient direction during iterations. In each iteration, the gradient will be saved, and then combined using a decay
factor with the previously saved gradients. The work [48] also demonstrated that by combining these gradients together, the gradient direction will be much more stabilized and the transferability of AEs could be enhanced. Furthermore, we also added random noise into the samples in each iteration to improve the robustness of the AEs, similar as in CommanderSong [105].

Specifically, let $g_{t+1}$ be the gradient in the $(t+1)th$ iteration, and $g_0$ be start gradient 0. Let $x_t^*$ denote the AE generated in the $(t)th$ iteration, and $x_0^*$ be original audio. $Clip_\varepsilon$ is a function clipping the values exceeding the pre-defined maximum and works in each iteration. Therefore, $x_{t+1}^*$ within the $\varepsilon$ vicinity can be obtained based on MI-FGM as below:

\[
g_{t+1} = \mu \cdot g_t + \frac{J(x_t^*, y)}{\|\nabla_x J(x_t^*, y)\|_1} \tag{4.1}
\]

\[
x_{t+1}^* = x_t^* + Clip_\varepsilon (\alpha \cdot g_{t+1}) \tag{4.2}
\]

where $y$ is the probability value of the target pdf-id sequence of $x_t^*$, $\mu$ is the decay factor for the momentum, $\alpha$ is the step factor\(^5\), $J(x_t^*, y)$ is the loss function. Intuitively, MI-FGM uses the gradient of the loss function to determine the direction, along which

\(^5\)Dong et al. evaluated the success rate of AEs for different decay factors and found 1.0 is the optimal value [48]. Carlini et al. used Adam optimizer to minimize the loss function where the default step factor $\alpha$ is set as 100 [39]. In this paper, we set those two factors based on the above two works.
the loss function itself can be minimized. Compared with normal FGM, MI-FGM replaces the current gradient with the accumulated gradients of all previous steps.

Based on our evaluation, the enhanced approach helps to generate a few AEs attacking black-box ASR API services (e.g., Google Cloud Speech-to-Text API) with low success rate and works even poorer on IVC devices (see Section 4.6.2). The main reason is that the approach to generate AEs mainly depends on the sample’s transferability to other black-box systems. Thus, we consider the transferability based approach has one major limitation: the crafted AEs are generated more towards the white-box model. However, the decision boundaries may vary between the white-box model used to generate the AEs and the target black-box model.

4.3.2 Alternate Models based Generation

We introduce the alternate models based generation approach to effectively hack the commercial black-box API service systems and IVC devices.

Approach overview. First, we propose to build our carefully augmented corpus to train a local model approximating the target black-box model on the desired commands. As the AEs generated from Kaldi ASpIRE Chain Model can be transferred to the target black-box model in some extent, we take it as the large base model, and use it to enhance the small substitute model to generate the AEs. Therefore, the large base model can generate most of the acoustic features of the desired command (Step ① in Figure 4.1). Furthermore, the last generated AE of the base model will be fed into the
substitute model (Step 2 in Figure 4.1). Thus, the unique features of the desired command on the target model can be adjusted in a fine-grained manner by the substitute model (Step 3 in Figure 4.1), since it was trained based on an augmented corpus (details in Section 4.3.2.1) that can be well recognized by the black-box model. During the AE generation process under each model, we use a small subset of AEs to query the target ASR API service according to our query reduction method (Step 5 and Step 6 in Figure 4.1). If none of these AEs works, the last crafted audio (an unsuccessful AE) from the substitute model will be fed to the base model as the input for the next epoch (Step 4 in Figure 4.1). Finally, we select the effective AEs to compromise the target IVC devices (Step 7 in Figure 4.1). Below we detail such approach.

4.3.2.1 Local Model Approximation

Here we discuss how we prepare the local model approximation.

**Training set with limited number of phrases.** Generally, the commercial black-box models are trained with significantly large proprietary dataset, and the structure of the neural network can be quite complicated. Therefore, it is extremely difficult to obtain the corpus or even infer the details about neural network. In other words, training a local substitute model completely approximating the target commercial black-box system is almost unpractical. However, since our ultimate goal is to hack the commercial IVC devices and in turn leverage it to compromise the victim’s digital life, we are only interested in a limited number of selected phrases such as “open my door”, “clear
notification”, etc. A side product of selecting those phrases is that, based on our experiences, the IVC devices are trained to be quite robust to those phrases, e.g., “open my door” on Amazon Echo, “what is the weather” and “play music” on Microsoft Cortana and Google Assistant. Hence, we just need to train a local model partially approximating the target system on the most frequently used phrases, also the ones we are highly interested in, on IVC devices. We use Text-to-Speech services to generate TTS audio clips for our desired phrases (details in Section 4.5.3) as the training set for local model approximation.

**Training set augment.** The above observation inspired us the idea of the local partial approximation. However, the training set has two problems: the number of phrases in the training set is too limited for training; and the robustness of an IVC device to a phrase is unknown. To solve these problems, we augment the training set by tuning the TTS audio clips, i.e., either changing the speech rate of and adding noises to them. Based on our experience, the changing of the speech rate and the noise amplitude is quite unique to different ASR systems, e.g., a specifically tuned audio might be decoded correctly with high confidence by one ASR system, but incorrectly by the other. Hence, we believe that those tuned but still correctly decoded audio clips can help to uniquely characterize different ASR systems, and that training an ASR system with plenty of such audio clips will guide it towards the target ASR system on the desired phrases in the audio clips.
Obviously, not all the tuned audios can still be decoded correctly by the target black-box system. In our research, we assume that the speech recognition mechanisms of the IVC devices are similar to that of the API service provided by the same company\(^6\). Hence, we query the corresponding online speech API service on them, and filter out those either not correctly decoded, or decoded correctly but with low confidence values. The magnitude of the corpus augmented in this way is not very big, usually 3~6 hours for ten selected phrases, which can be finished in about 1500 queries on the target speech API service.

### 4.3.2.2 AE Generation

Next we discuss how we generate our AEs.

**Generating AEs with base model and substitute model.** After the local substitute model is trained based on the augmented dataset, we ensemble it with the large base model for the alternate models generation summarized in Algorithm 1. Specifically, Line 3 and Line 4 are for the AE generation on the large base model and the small substitute model respectively. The AE generation is the same for two models and defined as the function “AEGENERATION” in Line 8~24.

---

\(^6\)Although previous studies [69, 107] show that it is possible to recover Speech-to-Text functionality from some IVC devices like Echo, their approaches cannot obtain the confidence values for the decoded results, which are required in our approach.
**Algorithm 1** Alternate Models Generation Algorithm

**Require:** The original audio $x_0$, the target label $y$, $f_{target}$ is the function to get output from black-box model, the black-box query interval times $T_{interval}$, the maximum allowed epoch $EpochMax$.

**Ensure:** A set of adversarial example collection $X^*$, all with label $y$ under classifier $f_{target}$.

1: $x^*_0 = x_0 ; g_0 = 0 ; CurrentEpoch = 0 ; T^*_interval = T_{interval}$
2: **while** $CurrentEpoch < EpochMax$ **do**
3:  AEgeneration (Base Model Settings);
4:  AEgeneration (Substitute Model Settings);
5:  $CurrentEpoch++$;
6:  **end while**
7:  **return** $X^*$
8:  **function** AEGENERATION(Model Settings)
9:  Reset $T^*_interval = T_{interval}$;
10:  **for** each $t \in [0, T - 1]$ **do**
11:  Take $x^*_t$ for current model $f$ and get the gradient;
12:  Update $g_{t+1}$ by Eq. 4.1;
13:  Update $x^*_{t+1}$ by Eq. 4.2;
14:  **if** $t \mod T^*_interval = 0$ **then**
15:  Input $x^*_{t+1}$ to $f_{target}$ and get $f_{target}(x^*_{t+1})$;
16:  **if** $f_{target}(x^*_{t+1})$ match $y$ **then**
17:  Put $x^*_{t+1}$ into $X^*$;
18:  **else**
19:  Update $T^*_interval$ by Eq. 4.3;
20:  **end if**
21:  **end if**
22:  Set $x^*_0 = x^*_T$;
23:  **end function**

In each iteration of the **for** loop starting at Line 10, the gradient is updated in line 12 based on Eq. 4.1 and the audio sample is crafted in line 13 based on Eq. 4.2.

To successfully attack the target black-box model, we need to query the target speech API service and validate whether the decoded result of the crafted audio sample is as expected or not. An intuitive way is to submit the sample generated in each iteration,
so any potential effective AE will not be ignored. However, such method will incur a significant amount of queries sent to the API service, which could be costly and at the same time suspicious. Therefore, we implement the query reduction algorithm (will be detailed at the end of this subsection), which aims to reduces the number of queries to the target black-box system. At Line 1, we set the $T_{\text{interval}}$ as the number of iterations between two consecutive queries to the target black-box system, and then at Line 19, it is updated based on Eq. 4.3, according to the recognition results from the target black-box system.

If after $T$ iterations, effective AE is still not generated (i.e., Line 16 always returns false), we assume the transferability based attack does not work well towards the target black-box system. We will use the output $x^*$ from the last iteration as the input to the local substitute model, then use the same gradient algorithm to craft the adversarial sample under the substitute mode settings. If after we reach the $T$ iterations and the local substitute model approximation approach still does not succeed in generating the AE for the target command, we go back to Line 2 to restart the whole algorithm. The $\text{EpochMax}$ parameter can restrain the number of total alternations. For Line 16~17, we will not break the “AEGENERATION” function even if the Line 16 returns “True” and an effective AE is crafted towards the target ASR API service. This is because the successful sample to attack the target ASR API service does not necessarily indicate the success towards IVC devices. Therefore, instead of breaking the function, once a successful AE is found, we save it towards the target ASR API service. Finally, we
can return a set $X^*$, where we preserve all potential effective AEs towards target IVC devices.

**Efficient query of the black-box API.** Intuitively, we can query the black-box server after a few iterations, instead of every iteration. We compare the decoded transcript of the current sample from the black-box model with the desired transcript, and use it as a reference to determine when the next sample should be sent to the black-box server. Suppose we set the number of iterations between two queries to the target black-box model as $T_{interval}$, and there are $s$ words from the decoded transcript of AE that match the desired commands (e.g., $s = 2$ if “the door” is decoded from the current iteration for the desired command “open the door”). Then $T_{interval}^*$ should be updated by Eq. 4.3.

$$T_{interval}^* = \left\lfloor T_{interval} \times \frac{1}{s + 1} \right\rfloor$$ (4.3)

Actually when examining the word match, we check the phonetic similarity of the words, rather than character-by-character match, since the language model of the speech recognition systems will refine the phonetic-similar words based on semantics. Hence, we applied SoundEx [93], a tool to encode homophones with the same representation even though they have minor differences in spelling. Thus, $s$ will be updated by comparing the SoundEx code of the decoded command and the target command. For example, “inter” is encoded by SoundEx as “I536” and “pay Internet fee”
is encoded as “P000 I536 F000”. We consider one match (the code “I536”) when comparing such decoded output and desired command, so s will be set as 1 in this case.

### 4.4 Understanding the Attacks

Although our approach works effectively in a black-box attack, which will be demonstrated in our experiments (Section 4.6.2), theoretic analysis of the technique are non-trivial, just like the attempt to interpret adversarial learning in general. Following we provide high-level intuition behind our approach through an example.

At a high level, our approach is based upon the observation that different ASR systems have some similarity in their classification models, which allows us to utilize a white-box, well-trained base model to move an instance towards the target model’s decision boundary, though it is likely different from that of the white-box model. This difference is further addressed using the substitute that fine-tunes the instance based upon the features of the target, including those related to its decision boundary. In this way, we can utilize both the information learnt from the target by the substitute and that shared between the base model and the target to build a successful AE.

For example, consider an attack on the Alexa Transcribe API using the approach we proposed previous section. The target command is “clear notification”. According to the experimental results, we found that the generation process (the base model-
substitute model-base model) helped find the results that came closer to the target (recognized as “I don’t” by Alexa Transcribe API). These results were then further adjusted by the substitute model towards the target. They became “notification” in the 10th~30th iterations, and were recognized as “clear notification” in the 48th~60th iterations. We believe that the transformation from “I don’t” to “clear notification” is attributed to the fact that the substitute is trained to simulate the behavior of the Alexa Transcribe API on “clear notification”.

4.5 Implementation

In this section, we introduce the implementation details of this attack, including which target systems and devices we choose, which target phrase we choose and how we conduct local model approximation.

4.5.1 Target IVC Devices and ASR Systems

Since we are developing a general approach generating AEs against commercial black-box IVC devices, we plan to examine the AEs on most of the popular IVC devices currently available on the market. In particular, we consider the speech recognition devices/systems from Google, Microsoft, Amazon, Apple, and IBM into the following three categories. First, we can find the ASR API services associated with the corre-
sponding IVC devices, e.g., Google Assistant and Google Cloud Speech-to-Text API\(^7\) (Category 1). Second, IVC device is available, but ASR API is not, e.g., Apple Siri (Category 2). Last, ASR API is available, but IVC device is not, e.g., IBM Speech to Text API (Category 3).

Regarding Category 2, since there does not exist online ASR API service required by local model approximation, we attack such IVC devices mainly via transferability as in Section 4.3.1. As for Category 3, since we cannot find the IVC device of IBM, we simulate such scenario by playing the AE, recording it and then using the ASR API service to decode the recorded audio as in Section 4.6.3. All the available ASR API services return the decoded transcripts and the corresponding confidence level for the decoding.

### 4.5.2 Phrase Selection

Since the aim of our approach is to attack the commercial IVC devices like Google Home, we only focused on the specific commands frequently used on these devices, e.g., “turn off the light”, “navigate to my home”, “call my wife”, “open YouTube”, “turn on the WeMo Insight”, etc. For each target model, we selected 10 such com-

\(^7\)There are four models in Google Cloud Speech-to-Text API, e.g., “phone call model”, “video model”, “command and search model” and “default model”. In detail, “phone call model” is used to translate the recorded audio from phone call; “command and search model” is used for voice command and short speech searching; “video model” is used for the video; “default model” is not designed for a specific scenario. We use the command and search model to label our corpus since our corpus are more suitable for voice command and search application.
mands and further appended the default wake-up words for different systems (Google Home, Amazon Echo and Microsoft Cortana) before each of them. For the IBM Speech to Text API without commercial IVC devices available, we utilized daily conversation sentences.

4.5.3 Local Model Approximation

We then discuss how we train our local approximation model.

Model selection. In our experiment, we chose the Mini Librispeech model\(^8\) as the substitute model to approximate the target models. Specifically, we used the default architecture and hyper-parameters of Mini Librispeech to train all four substitute models in our paper. These models were found to be highly effective in our study (Section 4.6.2). On the other hand, we acknowledge that even better performance could be achieved by tuning model parameters, a mostly manual and time-consuming procedure. So, our attack should only be viewed as a lower bound for the security threats these commercial systems are facing.

Corpus preparation. To enrich our corpus, we use 5 TTS (Text-to-Speech) services to synthesize the desired command audio clips, i.e., Google TTS [13], Alexa TTS [1], Bing TTS [7], IBM TTS [17] and an unnamed TTS [10], with 14 speakers in total including 6 males and 8 females. After using the above TTS services to gen-

---

\(^8\)Both Mini Librispeech and Kaldi ASpIRE (used as the base model) use chain model, and Mini Librispeech is easy to implement.
erate the desired command audio clips, we enrich it by adding background noise or twisting the audio. For the former, we add white noise to the original audio, and set the amplitude of the added white noise to be $\alpha$. For the latter, we twist the original audio by changing its speech rate either slower or faster. We define the twist-rate as $\beta$ ($\beta = \frac{\text{original\_audio\_duration}}{\text{twisted\_audio\_duration}}$). Finally, we use the target black-box model to recognize the tuned audio and filter it based on the correctness and the confidence level of the decoded results.

We constructed the training corpus by combining the tuned TTS audio clips (generated from the queries on the target model) and the supplemental corpus from Mini Librispeech. This is because the tuned TTS audio clips alone would cause the substitute model to overfit to the set of commands used in the queries (in the tuned TTS audio clips). As a result, the AEs found from the less generalized substitute model can be less effective at attacking the target models. On the other hand, solely relying on the supplemental corpus is not effective either, since the substitute trained without the information from the target will behave very differently from the target, as confirmed by our experiment (alternate models based generation without approximation) in Section 4.6.4.

Furthermore, we evaluate the impact of different sizes of supplemental corpus on Microsoft Bing Speech Service API, and the results show that 3-40 hours size of the supplemental corpora are all effective for our approach, while with 1 hour supplemental data cannot generate AEs for all of the target commands. For the four substitute
models of the target black-box platforms, we use the default Mini LibriSpeech corpus (7.35 hours) as the supplemental corpus.

**Training the substitute model.** To train the substitute model, we need to label the audio clips in the training corpus. Also, as mentioned in Section 4.3.2, retrieving the pdf-id sequence of the target commands is critical in our attack. However, we found that some words (such as Cortana, WiFi, Bluetooth, YouTube, WeMo, TV, etc.) are not included in the dictionaries of the Mini Librispeech model and the ASpIRE Chain model, so we cannot directly label these words and get the pdf-id sequences of the corresponding commands. Simply extending the vocabulary of the language models [22] requires the entire language models be retrained. To address this problem, we leveraged some linguistically similar phrases, based upon the prior research [69, 107], to label those undocumented ones\(^9\), which allows us to identify the pdf-id sequences of their commands and further generate their AEs.

### 4.6 Evaluation

In this section, we show the experimental results of Devil’s Whisper attack.

#### 4.6.1 Experiment Setup

First, we show the setup details of our experiments.

\(^9\)The phrases like “Cortana”, “why fi”, “blue tooth”, “you too boo”, “we mow” and “T V” are used to replace “Cortana”, “WiFi”, “Bluetooth”, “YouTube”, “WeMo” and “TV”, respectively.
**Hardware.** We conduct the experiment on the sever equipped with four Nvidia Tesla K40m GPUs and 2 x 10 core Intel Xeon E5-2650 2.30GHz processors, with 131 Gigabytes of RAM and 1 Terabyte Hard Drive. We use a laptop (Lenovo W541/Dell XPS 15/ASUS P453U) and a phone (iPhone SE/iPhone 8) connected to a speaker (JBL clip 2/3 portable speaker) to play out AEs. The target IVC devices are Google Home Mini, Amazon Echo 1st Gen and voice assistants on phones (Google Assistant App on Samsung C7100/iPhone SE and Microsoft Cortana App on Samsung C7100/iPhone 8). The transferability of the AEs on Apple Siri is tested on iPhone 8 and iPhone XR. The AEs on IBM WAA tests are recorded by Huawei P30.

**The original audio.** Similar to CommanderSong, our attack utilizes songs as the carrier for the AE produced. Specifically, we used the dataset released by the CommanderSong project [9], which contains 5 songs in each of the soft, popular, rock and rap categories. Among them, we selected the songs in the soft and popular categories, which are less noisy, allowing the integrated perturbations more likely to overwhelm the background music and be decoded correctly by the target IVC devices. To further evaluate the 10 songs, we utilized two commands “Okay Google, navigate to my home” and “Hey Cortana, turn off the bedroom light”, and ran our approach to embed the commands into the songs, against the speech recognition APIs provided by Google and Microsoft Bing. The results show that all the 10 songs can serve as carriers for the commands to ensure their recognition by the APIs. However, when listening to these AEs, we found that four instances using soft songs and one using a popular song were
less stealthy than the other 5 manipulated songs and therefore selected the latter for all follow-up experiments. Our experimental results show that for each target command of each target platform, there are at least 2 music clips across these 5 songs that can be crafted as effective and stealthy AEs. Further we studied the songs more likely to be good candidates for covering different commands (Section 4.6.6).

Besides the songs, we also tried other types of sounds as our carriers for malicious commands in the experiments, e.g., ambulance siren sound, train passing sound, etc. We found songs perform best in both effectiveness and stealthiness among those sounds. Therefore, we choose the songs as our carrier.

### 4.6.2 Effectiveness

We evaluate the effectiveness of AEs generated by transferability based approach (TBA) and those generated by alternate models generation approach (AGA) on the commercial Speech API services and IVC devices. The target commands for every black-box platform are listed in Appendix of paper [41].

Similar to the existing works [39, 105, 90], we use $SNR^{10}$ to measure the distortion of AE to the original song.

---

$SNR$, defined as the ratio of the original signal power to the noise power, can be expressed as follows: $SNR(dB) = 10 \log_{10} (P_x(t)/P_\delta(t))$, where $P_x(t)$ represents the average power of the original signal and $P_\delta(t)$ represents the average power of the distortion. It can be seen that a larger $SNR$ value indicates a smaller perturbation.
**Speech-to-Text API services attack.** We feed our adversarial examples (AEs) directly into the corresponding API services, and observe the results. For the four models of Google Cloud Speech-to-Text API (Section 4.5), we show the results of “phone_call model” and “command_and_search model”, since according to our tests the former is similar to “video model” and the latter is similar to “default model”.

When attacking Speech-to-Text API, since we do not need to wake up the IVC devices, we consider the AE successfully attacks the target if the returned transcript matches the desired command. The results are shown in Table 4.1, with the $SNR$ being the average of all commands on each black-box platform. Specifically, the effectiveness of our approach is evaluated using the success rate of command (SRoC), that is, the number of successful commands vs. the total number of the commands evaluated on a target service. Here a successful command is the one for which we can generate at least one workable AE using our approach. The results show that the AEs produced by TBA work well on Google phone_call model with 100% SRoC, but fail on Google command_and_search model and Amazon Transcribe. Also the AEs generated by AGA achieve an SRoC of 100% for all Speech-to-Text API Services except Amazon Transcribe.

As for Amazon Transcribe API service, we only crafted successful AEs on 4 out of 10 target commands using AGA method. We then performed more tests on Amazon Transcribe API and found that the API service cannot even recognize some plain TTS audio clips for the target commands correctly. In contrast, these commands can always
Table 4.1 The overall SRoC results on API services.

<table>
<thead>
<tr>
<th>Black-box</th>
<th>Google</th>
<th>Microsoft Bing</th>
<th>Amazon Transcribe</th>
<th>IBM STT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Phone</td>
<td>Command</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TBA</td>
<td>2/10</td>
<td>1/10</td>
<td>3/10</td>
<td></td>
</tr>
<tr>
<td>AGA</td>
<td>10/10</td>
<td>10/10</td>
<td>10/10</td>
<td>10/10</td>
</tr>
<tr>
<td>SNR (dB)</td>
<td>11.97</td>
<td>9.39</td>
<td>13.36</td>
<td>11.21</td>
</tr>
</tbody>
</table>

Note: (1) “Phone” and “Command” represent the “phone_call model”, “command_and_search model” of Google Cloud Speech-to-Text API, respectively. (2) “Microsoft Bing” represents the Microsoft Bing Speech Service API. (3) “IBM STT” represents the IBM Speech to Text API. (4) The results were all based on the tests conducted in October 2019.

be recognized by Amazon Echo. There can be reasons for such difference. First, different models could be used by Amazon Transcribe API and Echo device. Second, the developers of Amazon Echo may set lower threshold to identify voice commands, thus it is more sensitive to the voice commands when used physically.

**IVC devices attack.** We selected the AEs that can successfully attack the API service with high confidence score ($\geq 0.6$) to attack the IVC devices. Specifically, since the AEs working poorly on Amazon Transcribe API are not necessarily working poorly on Amazon Echo as we identified before, we decide to test the AEs on Amazon Echo directly, even if they failed on Amazon Transcribe API. In our experiment, if the devices respond to the played AE in the same way as the regular voice command from human being, we consider the AE for this command successful.

As shown in Table 4.2, the average SRoC of TBA is 26%. In contrast, the average SRoC of AGA over all IVC devices can be improved to 98%, which shows the proposed approach is very effective in attacking real-world IVC devices. Based on our
evaluation, we find that for most of the black-box models, we can always find the AEs that can successfully attack their corresponding IVC devices from the ones that have fooled the ASR API services. However, Amazon Transcribe API and Amazon Echo are the exception. We find that although attacking Amazon Transcribe API is difficult, we can always generate AEs with 100% SRoC for the 10 target commands to attack Amazon Echo.

**Table 4.2** The overall SRoC results on IVC devices.

<table>
<thead>
<tr>
<th>Black-box</th>
<th>Google Assistant</th>
<th>Google Home</th>
<th>Microsoft Cortana</th>
<th>Amazon Echo</th>
<th>IBM WAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA</td>
<td>4/10</td>
<td>4/10</td>
<td>2/10</td>
<td>0/10</td>
<td>3/10</td>
</tr>
<tr>
<td>AGA</td>
<td>10/10</td>
<td>9/10</td>
<td>10/10</td>
<td>10/10</td>
<td>10/10</td>
</tr>
<tr>
<td>SNR (dB)</td>
<td>9.03</td>
<td>8.81</td>
<td>10.55</td>
<td>12.10</td>
<td>7.86</td>
</tr>
</tbody>
</table>

**Note:** (1) “WAA” is used to represent “Wav-Air-API” attack. (2) The results were all based on the tests conducted in October 2019.

We used a digital sound level meter “SMART SENSOR AS824” to measure the volume of AEs. The background noise was about 50 dB, and the played audios were about 65~75 dB, compared to some special cases of the sound level presented in [25], e.g., talking at 3 feet (65 dB), living room music (76 dB). We also conducted experiments to test our AEs in realistic distance. For example, the AE with the command “Echo, turn off the light” can successfully attack Echo as far as 200 centimeters away, and the AE with the command “Hey Cortana, open the website” can successfully attack Microsoft Cortana as far as 50 centimeters away.
Robustness of the attack. To evaluate the robustness of our attack, we define the success rate of AE (SRoA) as the ratio of the number of successful tests to the total number of tests if an AE has been repeatedly played. The results show 76% (38/50) of the commands have SRoAs over 1/3, showing that our attack is quite robust.

4.6.3 Attacking Other Platforms

We than evaluate our results on other platforms.

Over-the-air attack against IBM Speech to Text API. As stated in Section 4.5.1, we use “Wav-Air-API” (WAA) to simulate the IVC device of IBM. The results are shown in Table 4.2. Overall, such WAA attack demonstrates similar performance as other IVC devices, which further indicates the effectiveness and generality of our proposed approach.

AEs attack against Apple Siri. Since there is no online speech-to-text API service available from Apple, we tried two methods to attack Apple Siri: (1) we generate AEs directly using the transferability based approach; (2) we “borrow” the AEs demonstrating good performance on the other IVC devices. As shown in Table 4.3, only the command “What is the weather?” generated from TBA can attack Apple Siri successfully. For the other commands, we rely on the help from AEs generated from AGA for
other IVC devices\textsuperscript{11}. From Table 4.3, we find all the seven AEs can successfully attack Siri, which demonstrates the transferability of AGA\textsuperscript{12}.

\textbf{Table 4.3} Transferability of the Devil’s Whispe AEs on Apple Siri.

<table>
<thead>
<tr>
<th>Command</th>
<th>Black-box</th>
<th>TBA/AGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call 911.</td>
<td>Google</td>
<td>X/✓</td>
</tr>
<tr>
<td>Play music.</td>
<td>Google</td>
<td>X/✓</td>
</tr>
<tr>
<td>Set an alarm on 8 am.</td>
<td>Google</td>
<td>X/✓</td>
</tr>
<tr>
<td>Navigate to my home.</td>
<td>Google</td>
<td>X/✓</td>
</tr>
<tr>
<td>Turn on airplane mode.</td>
<td>Google</td>
<td>X/✓</td>
</tr>
<tr>
<td>What is the weather?</td>
<td>Microsoft</td>
<td>✓/✓</td>
</tr>
<tr>
<td>Call my wife.</td>
<td>Amazon</td>
<td>X/✓</td>
</tr>
</tbody>
</table>

\textbf{4.6.4 Evaluation of Possibly Simple Approaches}

In this subsection, we will discuss other potential simple approaches which can achieve the same goals as our attack.

\textbf{Local model approximation with a larger corpus.} Apparently, if the local model is trained by a larger corpus of tuned TTS audio clips, it could approximate the target black-box model better (Certainly a larger corpus means a larger amount of queries to the online API service, which could be suspicious.). Below we describe a preliminary evaluation of the AEs generated by such local model.

\textsuperscript{11}When testing AEs from the other IVC devices on Apple Siri, we ignore the wake up words, e.g., “OK Google, play music” should be truncated to “Play music”.

\textsuperscript{12}The test was conducted in January 2019. However, we found that the AEs cannot work on Apple Siri since July 2019 (details in Section 4.7).
We choose Google command_and_search model as our target system. Then we pick up four commands that the AEs generated by our approach can be decoded by Google command_and_search model with 100% SRoC. As in Section 4.3.2.1, we use TTS to generate regular speech of those commands, and extend the corpus by tuning TTS audio clips. Finally the corpus is filtered out by the labeling from Google command_and_search model with the same confidence level as that in our approach. Hence, we obtain a corpus of about 23.86 hours (5100 oracle queries), almost 5.17 times larger than that used in our approach. After the local model is trained with the larger corpus, we use the “MLFGM” algorithm to generate AEs and evaluate them on the target.

The results show only one command “OK Google, turn off the light” succeeds on Google command_and_search model, but still fails on Google Home. The other commands do not have any successful AEs generated for Google command_and_search model and Google Home/Assistant. Based on the results of the preliminary testing, even if the adversary could afford the cost of preparing larger corpus and a larger amount of queries, the AEs generated from such simplified approach is not as effective as our proposed alternate models based generation with approximation approach.

**Alternate models based generation without approximation.** Another intuitive approach is based on the assumption that if one AE works on multiple models, it is highly possible that it works on the target model, without the need to approximate the target. We kept the ASpIRE Chain model as the base model, and trained the Mini
Librispeech model without the tuned TTS corpus. Specifically, we selected four target commands to attack Google command_and_search model and Google Assistant/Home. We ran the proposed alternate models based generation approach based on those two models (ASpiRE Chain model and Mini Librispeech model) to craft AEs. However, only one out of four commands works on Google command_and_search model and Google Assistant, while all the four commands fail on Google Home.

**Table 4.4** Results of the comparison tests with different approaches.

<table>
<thead>
<tr>
<th>Black-box</th>
<th>Target command</th>
<th>Original song + TTS</th>
<th>Devil’s Whisper</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SRoA of Group1</td>
<td>SNR (dB)</td>
</tr>
<tr>
<td>Google Assistant</td>
<td>Okay Google, take a picture.</td>
<td>6/10</td>
<td>7.15</td>
</tr>
<tr>
<td></td>
<td>Okay Google, navigate to my home.</td>
<td>3/10</td>
<td>4.08</td>
</tr>
<tr>
<td>Google Home</td>
<td>Okay Google, turn off the light.</td>
<td>6/10</td>
<td>4.05</td>
</tr>
<tr>
<td></td>
<td>Okay Google, play music.</td>
<td>2/10</td>
<td>4.53</td>
</tr>
<tr>
<td>Microsoft Cortana</td>
<td>Hey Cortana, open the website.</td>
<td>6/10</td>
<td>0.21</td>
</tr>
<tr>
<td></td>
<td>Hey Cortana, make it warmer.</td>
<td>9/10</td>
<td>3.38</td>
</tr>
<tr>
<td>Amazon Echo</td>
<td>Echo, turn off the computer.</td>
<td>6/10</td>
<td>3.39</td>
</tr>
<tr>
<td></td>
<td>Echo, call my wife.</td>
<td>4/10</td>
<td>4.78</td>
</tr>
</tbody>
</table>

**Note:** (1) The success rate “A/B” indicates that there are A tests success to trigger the command on the black-box platforms in B tests. (2) The results were all based on the tests conducted in July 2019. (3) Hardware settings: we used ASUS P453U as the audio source and JBL Clip 2 as the speaker for all test cases. Google Assistant and Microsoft Cortana were tested on Samsung C7100. Amazon Echo and Google Home were tested on Echo 1st gen and Google Home Mini. Volume of AEs is about 70 dB and distance ranges 5–15 centimeters.

**Other straightforward approaches.** We conducted experiments to compare our Devil’s Whisper attack with other straightforward approaches, i.e., “Plain TTS”, the AEs of CommanderSong, the “Original song + TTS”. Specifically, we selected eight target commands frequently used on four IVC devices, as shown in Table 4.4. Each command was covered by the same original song for different cases. Particularly, samples in “Original song + TTS” were generated by combining the song and the TTS command with Adobe Audition software. Note that for such a simple combination, whether the injected command can be clearly heard and therefore interpreted by
the IVC depends heavily on the strength of the signal from the song (in terms of its volume) vs. that of the command in the TTS audio. To evaluate the perturbation of the TTS audio on the original song, we calculated the *SNR* of the combinations by treating the TTS audio (the command) as noise and the song as signal.

The results of our experiment are shown in Table 4.4. Overall, the AEs from the Devil’s Whisper attack can effectively attack the target IVC devices using those commands. Without any surprise, the “Plain TTS” audios triggered the devices to act on those commands each time. The AEs produced by CommanderSong, which is not designed for the black-box attack, failed to achieve a single success on these devices. As stated in Section 4.3.1 under “Initial try”, sometimes CommanderSong AEs can be partially recognized as “Hey Google”, thus waking up Google Assistant/Home. Occasionally, part of the commands can be recognized by a woken Google Assistant or a Microsoft Cortana. However, none of the AEs (with the *SNR* between 2 and 14) could cause the IVC devices to act on the injected commands.

To produce the samples of “Original song + TTS” case, we set the volume of each TTS audio clip (the command) to the same level as in “Plain TTS” case, while adjusting the volume of the song as follows: (1) to achieve a similar success rate (SRoA) as our attack AEs (see the column in Table 4.4 under Group 1), and (2) to keep a similar *SNR* level as the AEs (Group 2). As we can see from the table, under a similar SRoA, all except one combined audio clips (Group 1) have much lower *SNR* levels compared with our AEs, indicating that the commands they include are likely
to be much more perceivable and thus much less stealthy, which has been confirmed in our user study (see Section 4.6.5, Table 4.5). The only exception is featured by a similar $SNR$ as our AE. When tuning the $SNR$ to a level of our AEs, we can see that the SRoA of most samples (all except one) go down to zero (Group 2). Also interestingly, even though the SRoA of our AE apparently is below that of the “Original song + TTS” audio clip for the command “Ok Google, take a picture”, we found that 60% of human users in our study could identify the hidden command, compared with 0% for our AE.

4.6.5 Human Perception

$SNR$ describes the relative strengths between signal and noise, which is traditionally used to measure the perturbation to data (e.g., an image) [43]. Naturally, it can also model the distortion to the song caused by an AE (with the song being signal and the command being noise), and therefore gives an intuitive and rough estimate of the AE’s stealthiness: the smaller $SNR$ is, the larger distortion to the song is imposed, so the more likely the source of the distortion – a hidden command can be perceived by human. This is largely in line with the findings from our user study as below. However, the metric will be less accurate, for example, when the distortion fits well in other background noise, becoming less easy to notice, even when the $SNR$ is low. In general, human perception of hidden commands is complicated, depending on individuals’ experience, the context of a conversation, etc. Finding an accurate measurement is still
an open question. Therefore, we conducted a survey\textsuperscript{13} on Amazon Mechanical Turk to evaluate human perception of the AEs generated by the Devil's Whisper attack, and compare the result with that of “Original song + TTS”. Specifically, we used the audio clips in Group 1 since they have the similar SRoA as Devil’s Whisper when attacking the target models.

The results of this user study are shown in Table 4.5. Here, the column “Normal” shows the percentage of the users who consider a sample to be normal music, and the column “Noise” gives the percentage of the users who find noise in songs. The column “Once-recognize” and the column “Twice-recognize” describe the percentages of the users able to recognize over half of the hidden command words\textsuperscript{14} after listening to the audio once or twice, respectively. As we can see from the table, 16.1% participants think that somebody is talking in the background when they listen to Devil’s Whisper, but nobody could recognize any command when an AE was played to them. By comparison, over 93% of the participants think that someone is talking when listening to the audio clips in “Original song + TTS”, and nearly 42.9% of them recognizes over half of the command words first time when they listened. Even if the participants were exposed to the same AEs for the second time, only 1.4% of them could tell over 50%

\textsuperscript{13}This survey will not cause any potential risks to the participants, such as psychological, social, legal, physical, etc. We do not ask any confidential information about the participants in the questionnaires. The IRB Exempt certificates were obtained from our institutes and can be found in https://github.com/RiskySignal/Devil-Whisper-Attack.

\textsuperscript{14}We assume that 50% of the words in the command would be enough to raise user’s attention.
Table 4.5 Results of the human perception evaluation on Devil’s Whisper and original song combined with TTS command.

<table>
<thead>
<tr>
<th>Black-box</th>
<th>Approach</th>
<th>Normal (%)</th>
<th>Noise (%)</th>
<th>Talking (%)</th>
<th>Once-recognize (%)</th>
<th>Twice-recognize (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Assistant</td>
<td>Devil’s Whisper</td>
<td>14.3</td>
<td>74.3</td>
<td>11.4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Song and TTS</td>
<td>7.1</td>
<td>2.9</td>
<td>90</td>
<td>37.1</td>
<td>64.3</td>
</tr>
<tr>
<td>Google Home</td>
<td>Devil’s Whisper</td>
<td>14.3</td>
<td>65.7</td>
<td>20</td>
<td>0</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>Song and TTS</td>
<td>1.4</td>
<td>2.9</td>
<td>95.7</td>
<td>61.4</td>
<td>77.1</td>
</tr>
<tr>
<td>Microsoft Cortana</td>
<td>Devil’s Whisper</td>
<td>15.7</td>
<td>64.3</td>
<td>20</td>
<td>0</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>Song and TTS</td>
<td>2.9</td>
<td>1.4</td>
<td>95.7</td>
<td>31.4</td>
<td>54.3</td>
</tr>
<tr>
<td>Amazon Echo</td>
<td>Devil’s Whisper</td>
<td>25.7</td>
<td>61.4</td>
<td>12.9</td>
<td>0</td>
<td>2.86</td>
</tr>
<tr>
<td></td>
<td>Song and TTS</td>
<td>0</td>
<td>5.7</td>
<td>94.3</td>
<td>41.4</td>
<td>62.9</td>
</tr>
<tr>
<td>Average</td>
<td>Devil’s Whisper</td>
<td>17.5</td>
<td>66.4</td>
<td>16.1</td>
<td>0</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>Song and TTS</td>
<td>2.9</td>
<td>3.2</td>
<td>93.9</td>
<td>42.9</td>
<td>64.7</td>
</tr>
</tbody>
</table>

Note: (1) “Song and TTS” is used for the abbreviation of “Original song + TTS”. (2) “Once-recognize” and “Twice-recognize” represent that the users can recognize over half of the hidden command when they listen to the AEs for once and twice, respectively.

words in the target commands in the Devil’s Whisper attack, while the ratio goes up to 64.7% in “Original song + TTS”. This indicates that, the samples from “Original song + TTS” are much more perceptive to users. Furthermore, by analyzing the $SNR$ in Table 4.4 and human perception results, we found that $SNR$ was largely in line with human perception but not always (see the exception described in Section 4.6.2). The details of the survey can be found in https://github.com/RiskySignal/Devil-Whisper-Attack.
4.6.6 Selection of Songs

In order to find what types of songs are good candidates for our attack in terms of both effectiveness and stealthiness, we conducted a preliminary evaluation using all the 20 songs from CommanderSong, including the 5 rock and 5 rap songs that we did not use in our attack (see Section 4.6.1). The target commands were the same as those used in the previous experiments for Google and Microsoft Bing. In the evaluation, a song is considered to be suitable for AE generation if it helped produce effective AEs (for both commands) in the first epoch (based model - substitute model). Note that an effective AE is stealthy, as determined by humans (authors and other group members in our research) who listened to it. Through the evaluation, we classified the 20 songs into three categories: (1) easy to generate successful AEs but noticeable to human (2) easy to generate successful AEs and unnoticeable to human (3) hard to generate successful AEs. Obviously, the songs in the second category are good candidates for our attack. These songs are characterized by the similarity in the energy distributions of their spectra, as discovered in our research. We here present an example to show its spectral power distribution in Figure 4.2, where spectrum (a) represents Type 1: easy to be generated as successful AEs and perceived by human, spectrum (b) represents Type 2: easy to be generated as successful AEs but difficult to be perceived by human and spectrum (c) represents Type 3: hard to be generated as successful AEs.
Further we looked into the Top 100 Billboard songs in the week of 11/04/2018, embedding the commands “Hey Cortana, what is the weather?” (Command A) and “Hey Cortana, make it warmer” (Command B) into each of them, in an attempt to attack the Microsoft Bing Speech Service API, and “Ok Google, turn off the light” (Command C) and “Ok Google, navigate to my home” (Command D) to attack the Google Cloud Speech-to-Text API. During the attack, we selected the segment between the 60th second to the 63rd second (roughly the middle of the songs) for each song as the carrier for the commands. For Command A, B, C, D, we successfully generated AEs based on 59, 56, 58 and 60 songs, respectively. Then we asked 20 students to listen to the successful AEs generated and reported the commands that could be recognized. In the end, again, we classified all the 100 songs into these three categories. Most of their frequencies and energy distributions were found to be in consistent with those discovered in the 20 songs (see the example in Figure 4.2). This indicates that indeed
a more systematic way to select ideal carriers for the attack is possible, which will be explored in the future research.

4.7 Limitations

It is known that AEs are rather sensitive to the change made on the deep neural network models behind ASRs: even a small update could cause a successful AE to stop working. This is also true for our approach. For instance, the previous workable AEs (in January 2019) cannot work effectively towards Apple Siri since July 2019 (See Section 4.6.3).15 A potential solution is to fine-tune the existing model in the hope of capturing the impact of the change, which will be studied in the future research. In addition, the practical attack against IVC devices is sensitive to various environmental factors, such as the volume when playing AEs, the distance between the speaker and the IVC device, even the brand of the speakers, etc., which may significantly affect the SRoA. Hence, how to improve the robustness of the AEs in diverse environments is still an open research question. Finally, although user study shows that none of the participants can identify any command from our AEs if they only listen to them once, a few participants felt our AEs noisy/abnormal. Therefore, improving the stealthiness of AEs is on demand.

15 We further used eight samples of the case “Original song + TTS” from Table 4.4 to attack Siri and only 1 out of 8 samples can work. So, we consider that Siri may have updated the system to ignore the speech with music background.
4.8 Defense

We discuss three potential defense mechanisms to mitigate our Devil’s Whisper attack.

**Audio downsampling.** Audio downsampling was proposed in [105] to effectively mitigate the AEs. Even though the audio can be recorded in different formats (such as m4a, mp3, wav) at different sampling rates (e.g. $8000\ Hz$, $16000\ Hz$, $48000\ Hz$), we can always first downsample it to a lower sampling rate and upsample it to the sampling rate that is accepted by the target black-box model. During such down-sampling/upsampling process, the added adversarial perturbations may be mitigated, which makes the AEs fail to be recognized by the target black-box model. For instance, we choose the recorded audios, which can succeed in WAA attack on IBM Speech to Text API. Then they are downsampled to $5600\ Hz$, and upsampled to $8000\ Hz$, which are sent to IBM Speech to Text API. Only 20% of them can be recognized as the target commands. When first downsampled to $5200\ Hz$ and then upsampled to $8000\ Hz$, none of them can succeed. In contrast, the regular recorded human voice and TTS audio clips can still be recognized correctly even after such downsampling/upsampling. Hence, audio downsampling could be one effective way in detecting speech AEs. However, if an attacker know the downsampling/upsampling rates of the defense, he could train an AE robust against it.

**Signal smoothing.** Since the effectiveness of our AEs is highly dependent on the carefully added perturbations by gradient algorithm, we can conduct local signal
smoothing towards AEs to weaken the perturbations. Specifically, for a piece of audio 

\[ x \]

we can replace the sample \( x_i \) with the more smooth value according to its local reference sequence, i.e. the average value of the \( k \) samples before and after \( x_i \). Hence, the added perturbations may be mitigated by this method.

**Audio source identification.** Audio source identification aims to identify the source of the audio, e.g., from an electronic speaker or human. Such defence is based on the assumption that the legitimate voice commands should only come from human rather than an electronic speaker. Therefore, if the audio is detected not from human, the audio signal will be simply ignored. Previous works [40, 54] show that they can identify the audio source by either examining the electromagnetic wave from the audio or training a model to label the audio. Such defence mechanism could work for most of the existing speech AEs that require a speaker to play. However, the attacker could play the samples over a long range, which might evade the detection.
CHAPTER 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, we systematically explore the vulnerability of current speech recognition systems and devices under audio adversarial attacks. We first conduct a white-box based audio adversarial attack towards open-source speech recognition toolkit Kaldi [21]. In this work, we show that by injecting “voice” commands into songs (CommanderSong), the target white-box ASR system could be compromised by the adversarial samples. Then, followed by the white-box attack, we further perform Devil’s Whisper, a general adversarial attack on commercial black-box ASR systems and IVC devices, and the adversarial examples are stealthy enough to be recognized by humans. The key idea is to enhance a simple substitute model roughly approximating the target black-box platform with a white-box model that is more advanced yet unrelated to the target. The two models are found to effectively complement each other for generating highly transferable and generic adversarial examples on the target. To
sum up, this thesis show that audio launching adversarial attacks is feasible for current ASR systems and IVC devices.

### 5.2 Future Work

Looking ahead, we believe that there are several exciting new research directions that remain to be explored in this area. First, we believe that optimization of selection of original songs in both white-box and black-box attack would be interesting and meaningful, which could potentially help to increase success rate and minimize the perceptibility of the samples. Additionally, we believe it would be significant and valuable to conduct research on how to improve the physical robustness of the adversarial samples in the real world, as this would possibly bring severe impact to real world users. Lastly, more types of attacks like backdoor attack towards deep learning based speech systems could also be an interesting research direction as there lacks such research yet in the community.
References


[8] CMU Sphinx. cmusphinx.github.io/.


[104] Xuejing Yuan, Yuxuan Chen, Aohui Wang, Kai Chen, Shengzhi Zhang, Heqing Huang, and Ian M Molloy. All your alexa are belong to us: A remote voice control attack against echo. In 2018 IEEE Global Communications Conference (GLOBECOM), pages 1–6. IEEE, 2018.

