Semi-black-box Attacks Against Speech Recognition Systems Using Adversarial Samples

Yi Wu*, Jian Liu*, Yingying Chen†, Jerry Cheng‡
*University of Tennessee, Knoxville, TN, USA
†Rutgers University, New Brunswick, NJ, USA
‡New York Institute of Technology, New York, NY, USA
Email: ywu83@vols.utk.edu, jliu@utk.edu, yingche@scarletmail.rutgers.edu, jcheng18@nyit.edu

***Abstract*—As automatic speech recognition (ASR) systems have been integrated into a diverse set of devices around us in recent years, security vulnerabilities of them have become an increasing concern for the public. Existing studies have demonstrated that deep neural networks (DNNs), acting as the computation core of ASR systems, is vulnerable to deliberately designed adversarial attacks. Based on the gradient descent algorithm, existing studies have successfully generated adversarial samples which can disturb ASR systems and produce adversary-expected transcript texts designed by adversaries. Most of these research simulated white-box attacks which require knowledge of all the components in the targeted ASR systems. In this work, we propose the first semi-black-box attack against the ASR system - Kaldi. Requiring only partial information from Kaldi and none from DNN, we can embed malicious commands into a single audio chip based on the gradient-independent genetic algorithm. The crafted audio clip could be recognized as the embedded malicious commands by Kaldi and unnoticeable to humans in the meanwhile. Experiments show that our attack can achieve high attack success rate with unnoticeable perturbations to three types of audio clips (pop music, pure music, and human command) without the need of the underlying DNN model parameters and architecture.**

***Index Terms*—semi-black-box attacks, adversarial samples, speech recognition, Kaldi, deep neural network**

I. Introduction

Nowadays automatic speech recognition (ASR) systems, such as Siri, Google, and Alexa Assistants, are ubiquitous in our daily lives. These systems provide human voice interfaces for the convenience of various tasks, such as access control, smart home appliance control, voice navigation, and information query. Increasingly many devices have embedded the ASR systems, including smartphones, standalone smart speakers and smart home/office appliances. In the meanwhile, the rapid deployment of the ASR systems bring upon concerns of the potential security vulnerability in them.

As the computational core of ASR systems, deep neural networks (DNNs) have sufficient power for modeling large vocabularies and for coping with practical interference (e.g., noises and accent) to perform robust speaker-independent speech recognition. However, recent studies [2], [4], [12], [14] show that DNNs are vulnerable to adversarial perturbations, which allow attackers to translate speech into any text desired by an adversary. For instance, studies [2], [4], [12] are proposed to use a genetic algorithm for black-box attacks[1] or gradient-descent algorithm for white-box attacks[2] to generate adversarial examples against DeepSpeech [8], an open source Speech-To-Text engine implemented using Tensorflow. However, these approaches can only construct adversarial examples against DeepSpeech with a simplified end-to-end deep learning structure, which has a relatively lower speech recognition performance and is less favorable compared with other ASR systems such as Kaldi [10]. Moreover, CommanderSong [14] mainly targets Kaldi system due to its popularity. It can embed any malicious command into regular songs, making people hear them as common music whereas Kaldi system would recognize them as a malicious command. However, CommanderSong requires the adversary to know all of the essential components (e.g., extracted features, deep learning models) in the ASR system, which is not always possible in practice.

Different from existing studies, this paper proposes a semi-black-box adversarial attack against Kaldi, which is one of the most popular ASR systems. The proposed attack only requires partial information (e.g., embedding results of DNNs and HMM states' mappings) of the ASR system rather than the full knowledge of the system's DNNs models (e.g. its structure, weights or parameters). Thus, the proposed attack could make the attack feasible even if the adversary only has access to some specific function interfaces in the system. Particularly, we first derive DNN embedding results from an original audio sample and intermediate results of malicious commands calculated from the language model (LM). By referring to the mapping between two results, we implement a genetic algorithm to construct adversary perturbations in order to make the adversary samples recognizable by the ASR system but unnoticeable to humans. The main contributions of our work are summarized as follows:

- We propose a semi-black-box adversarial attack against Kaldi system, in which the adversary only requires partial information (e.g., the mapping between HMM states and transitions, DNN embedding output) rather than

[1]Black-box attack does not require the adversary to have any internal knowledge about the DNN model.

[2]White-box attack requires the adversary to possess the knowledge of the underlying ASR system.

TABLE I: Example of words with their composing phonemes.

Word	Composing Phonemes
yes	y_B eh_I s_E
no	n_B ow_E
speech	s_B p_I iy_I ch_E
attack	ah_B t_I ae_I k_E

the whole inner model knowledge such as the specific structure, weights or parameters of the DNN model.

- We show that using genetic algorithm could appropriately construct adversarial perturbations against the ASR system (e.g., Kaldi) without having to know the parameter and architecture of the system's DNN model.
- Our experiments of embedding adversarial perturbations into pop music, pure music, and pure voice commands, demonstrate the effectiveness and flexibility of the proposed adversarial attack against Kaldi.

II. Related Work

A. Adversarial Attacks Against DeepSpeech

Most of the existing adversarial attacks [2], [4], [5], [12] are focused on attacking DeepSpeech, which is a state-of-the-art speech recognition system developed using end-to-end deep learning [8]. The throughout flow allows it to directly convert complex inputs (e.g. raw audio) to comprehensive transcription texts based on computations from a single DNN. As a result, this makes adversarial perturbations relatively easy to construct. Meanwhile, DeepSpeech is implemented using Tensorflow, which provides a suitable coding environment for adversarial machine learning algorithms such as gradient descent and fast gradient sign method (FGSM) [7]. For instance, targeted black-box attacks using the genetic algorithm are implemented in [2], [12], while a white-box attack is proposed in [4] using the gradient descent algorithm and FGSM. Houdini [5] proposes a novel flexible approach for final performance measures of intended tasks, but it still requires an adversary to know the gradient of the target model.

B. Adversarial Attacks Against Kaldi

To the best of our knowledge, CommanderSong [14] and *psychoacoustic hiding*-based approach [11] are the only two attack approaches against Kaldi. Both of these attacks are a white-box attack, where an adversary needs to have the inner knowledge of an ASR system. By using gradient descent or back propagation algorithms, they can embed hidden commandsin audio clips which could be recognized by the Kaldi system. Different from existing studies, we propose a semi-black-box adversarial attack against Kaldi, which only requires partial knowledge of the ASR system. Instead of using the gradient-dependent algorithm, we demonstrate a gradient-independent algorithm (i.e., genetic algorithm) is also feasible towards adversarial attacks against Kaldi.

III. Background of Kaldi

Kaldi is an open-source toolkit used for speech recognition based on DNN-HMM. Written in C++ and licensed under the Apache License v2.0 [10], it is one of the most popular ASR systems among research and major companies such as IBM [3] and Microsoft [13]. Compared with DeepSpeech, the DNN-HMM structure utilized by Kaldi is more complicated and traditional, resulting in better speech recognition accuracy. For instance, Kaldi provides a Word Error Rate (WER) of 4.28% whereas DeepSpeech gives 5.83% on Librispeech clean data [9].

TABLE II: Transition model in Kaldi.

Phoneme	HMM	Pdf-id	Transition-id	Transition
n_B	0	307	36045	0 to 1
			36046	0 to 2
n_B	1	1692	36449	self-loop
			36450	1 to 2
ow_E	0	2725	39583	0 to 1
			39584	0 to 2
ow_E	1	89	39707	self-loop
			39708	1 to 2

TABLE III: The decoding result of word "no".

Phoneme	Transition-id Sequence
n_B	36405_36449_36449_36449_36449_ 36449_36449_36449_36449_36449_
ow_E	39583_39707_39707_39707_39707_ 39707_39707_39707_39707_39707_

In linguistics, phonemes as the smallest unit to compose a word could distinguish one word from another. As shown in Table I, different words are composed of different phonemes. Therefore, the ASR system needs to determine these phonemes in order to translate speech into text. To confirm each phoneme in Kaldi, a series of transitions among three HMM states are used. Table II illustrates the relationship between HMM states, pdf-ids and transition-ids of two phonemes (i.e., *n_B* and *ow_E*). A transition-id is defined to index a specific transition between two HMM states of the same phoneme, and a sequence of them directly determines which phoneme can lead to the transcription text. A DNN model plays a role to calculate the probability of each HMM state at each frame of the audio input, using the pre-trained probability density function (pdf). Additionally, the pdf is indexed using pdf-id, and from the computation result of DNN, we can obtain the pdf-id sequence with the highest probability. After obtaining all possibilities, the language model (LM) will do the calculation accordingly to derive the transition-id sequence. The transition-ids are mapped with HMM states, along with pdf-ids.

Table III shows the transition-id sequence calculated by the LM. The audio input is a single word "no" spoken by a native male speaker. Although the transition-id is calculated from the likelihood of pdf-id, after we obtain the determined sequence of transition-id, we can retrodict the sequence of pdf-id referring to the transition model in Table II. For the phoneme n_B, the pdf-id sequence should be *[307, 1692, 1692, 1692, 1692, 1692, 1692, 1692, 1692, 1692]*, and the pdf-id sequence for phoneme ow_E should be *[2725, 89, 89, 89, 89, 89, 89, 89, 89, 89]*. Therefore, if the pdf-id sequence with the highest probability of audio input is same or close to the above pdf-id sequence, the audio input would be recognized as n_B and ow_E, which then lead to the word "no".

TABLE IV: Different transition-ids can represent the same stage.

phoneme	HMM	Pdf-id	Transition-id	Transition
ow_E	1	57	39705	self-loop
			39706	1 to 2
ow_E	1	89	39707	self-loop
			39708	1 to 2
ow_E	1	129	39709	self-loop
			39710	1 to 2
ow_E	1	134	39711	self-loop
			39712	1 to 2

Additionally, we find that different pdf-id sequences may lead to the same malicious command. Therefore, if the pdf-id sequence of the adversarial sample is close to one of them, it can be recognized as the malicious command by Kaldi. In the transition model of Kaldi, we find that different transition-ids and pdf-ids may be corresponding to the same phoneme, HMM state and the transition. Table IV shows the possible transition-ids and pdf-ids of the phoneme *ow_E*. For instance, both the pdf-id sequence *[2725, 89, 89, 89, 89, 89, 89, 89, 89, 89]* and *[2725, 89, 57, 57, 89, 129, 134, 129, 89, 134]* lead to the phoneme *ow_E*, even though they differ significantly from each other. With a specific transition-id sequence of the malicious command and the transition model, we can derive thousands of different pdf-id sequences that can be regarded as "malicious pdf-id sequence".

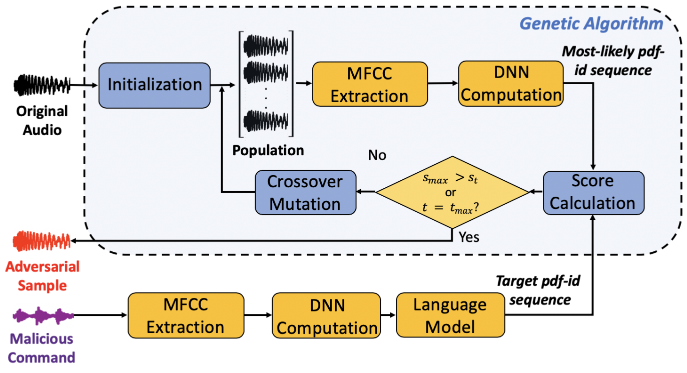

Fig. 1: Flow of the attack process.

Algorithm 1 Genetic algorithm

```
Input: Audio Input $X$, Malicious pdf-id Sequence $Y$
Output: Adversarial Sample $X'$
 1: function GENETIC($X, Y$)
 2:     $population \leftarrow Initialization(X)$
 3:     while $t < t_{max}$ and $s_{max} < s_t$ do
 4:         for each individual x in population do
 5:             $s \leftarrow ScoreCalculation(x, Y)$
 6:         end for
 7:         $bestSample \leftarrow population[argmax(scores)]$
 8:         $children \leftarrow Mutation(Crossover(population))$
 9:         $population \leftarrow bestSample + children$
10:         $MutationP \leftarrow newMuatationP$
11:         $t \leftarrow t + 1$
12:     end while
13:     return $bestSample$
14: end function
```

IV. Proposed Semi-black-box Attack

A. *Flow of the Proposed Attack*

The basic idea of the proposed attack is to generate an adversarial sample against Kaldi, which can make it produce the adversary-expected transcript text. As illustrated in Figure 1, the adversary first needs to take the original audio, which would be embedded with the malicious command, as the input. During *initialization*, the input audio would be added with different random nosies to obtain a *population* which contains various audio samples. For each individual in the population, the audio samples would go through *MFCC Extraction* to capture the MFCC features which will be taken as the input of DNN model. After *DNN Computation*, the *most-likely pdf-id sequence* for each individual will be derived, and then served as an input for *Score Calculation*.

Meanwhile, the malicious command also needs to be processed. Same as each individual in the population of the original audio, the command first needs to be processed through *MFCC Extraction* and *DNN Computation*. After obtaining the most-likely pdf-id sequence from the DNN, the *Language Model* will compute the *target pdf-id sequence* of the malicious command, which servers as the input of *Score Calculation*.

In *Score Calculation*, the most-likely pdf-id sequence for each individual will be calculated with the target pdf-id sequence to obtain a similarity score s. If the highest score in the population s_{max} is greater than the threshold s_t or the iteration time t reaches the max iteration time t_{max}, the attack ends and the individual with the highest score will be the final adversarial sample output. Otherwise, the attack will move to *Crossover and Mutation* to generate a new population to iteratively find the optimal adversarial sample.

B. *Generating Adversarial Sample based on Genetic Algorithm*

The genetic algorithm is a method used for solving optimization problems based on the principle of natural selection. It's commonly used to generate high-quality solutions by relying on bio-inspired operators such as mutation, crossover, and selection [6]. At each step, the genetic algorithm selects individuals who have a relatively better performance from the current population as parents and derive a child from them for the next population. For each generation, it only keeps the individual who has the highest score and aligns it with other children to be the next population. Over successive iterations, the population "evolves" towards an optimal solution. The algorithm ends when the iteration time t reaches the max iteration time t_{max} or the score s is high enough to reach the threshold s_t. It is instrumental in solving a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, non-differentiable, stochastic, or highly nonlinear. The pseudocode of the genetic algorithm used in our attack is stated in Algorithm 1.

Initialization. In particular, given an original input X, we generate $populationSize$ crafted audio samples by

TABLE V: An example of crossover mechanism.

Audio Array 1	[	-1696	-529	718	...	10054	9485	9914	]
		*	*	*	...	*	*	*	
Random Weights	[	0.4	0.6	0.3	...	0.2	0.9	0.7	]
		+	+	+	...	+	+	+	
Audio Array 2	[	-1580	-600	700	...	10050	9592	9923	]
		*	*	*	...	*	*	*	
Random Weights	[	0.6	0.4	0.7	...	0.8	0.1	0.3	]
		=	=	=	...	=	=	=	
Child Audio Array	[	1626	557	705	...	10051	9496	9917	]

adding different noises in it. The noises are different Gaussian white noises following the same normal distribution. $populationSize$ is the size of the population for each generation, which is a constant integer.

Score Calculation. For each generation, we calculate the similarity score s between each individual in the current population and the malicious command. Particularly, we calculate the Euclidean Distance from the individual's most likely pdf-id sequence to all of the possible pdf-id sequences of the malicious command, and then only keep the minimal value as L_{min}. And we set the score function as

$$s = e^{-L_{min}}. \tag{1}$$

The higher the score is, the more likely the crafted audio can be recognized as the malicious command, and we only keep the individual who has the highest score to be passed to the next generation.

Crossover and Mutation. *Crossover* and *Mutation* are two essential steps to generate children for the next population. The next population consists the amount of $populationSize$ - 1 children and the individual which has the highest score in the current population. Particularly, during *crossover*, two of the individuals in the current population will be chosen randomly as the parents to generate a child. Individual which has a higher score will more likely to be chosen. After selecting the two audio arrays a and b as the parents, we use the following formula to generate the child audio array c:

$$c_i = p * a_i + (1 - p) * b_i, \tag{2}$$

where p is the weight, which is a float randomly generated between 0 and 1. In this way, the child carries both parents' characteristics. Table V shows an example of crossover mechanism.

Note that if we only generate the children using *crossover*, after a certain number of iterations, all of the children will tend to be the same and will never produce better samples. *Mutation* is designed to overcome this issue. Each child has a probability of triggering mutation after being generated by crossover. If a child happens to trigger mutation, a relatively smaller Gaussian while noises will be added into it. We set *mutation_p* as the probability of triggering mutation for each child. After one iteration, if the highest score of the population doesn't increase (no better sample has been generated), we will increase the *mutation_p* slightly. Once the best score increases, we will set the *mutation_p* to its original value. We will keep iterating until one of the individuals' score reaches s_t or iteration time reaches t_{max}.

TABLE VI: Attack success rate.

Audio Type	Sample Number	Iteration	Success Rate
Pure Music	10	3000	90%
Pop Music	10	3000	70%
Voice Command	5	3000	20%

V. Performance Evaluation

A. Experimental Setup

In the experiments, we use three audio clips as the original audio input. They vary from a popular song "Good Time" by Owl City & Carly Rae Jepsen, a pure music "Take me hands" by Daishi Dance, to a normal voice command generated by the text speech engine [1] with a native male speaker sound. Similarly, we use TTSREADER [1] as the text-to-speech engine to generate various malicious commands. To generate adversarial audio samples, we use Google Cloud Compute Engine to run our proposed semi-black-box adversarial attack algorithm, in which the in-cloud virtual machine has 4 vCPUs with a memory of $16GB$. After constructing these adversarial samples, we directly feed the adversarial sample into Kaldi to obtain the speech recognition result.

B. Attack Success Rate

We consider an attack as success if the recognition result of an adversarial sample is exactly the same as the malicious command. For instance, we set the malicious command to be "yes" or "no", then even if the adversarial sample is recognized as "yeah" or "nope", it's also considered as a failure. The attack success rate of our attack is shown in Table VI. For pure music and pop music, we construct 10 adversary audio samples for each using the proposed genetic algorithm while for the pure voice command, we construct 5 adversary audio samples. Moreover, we set the max iteration time as 3000. Experiments demonstrate the effectiveness of the proposed semi-black-box adversarial attack against Kaldi ASR system, and the attack success rate can reach as high as 90%. It is important to note that if we expand the number of max iterations, we could further improve the attack success rate.

We observe that pure music has the highest attack success rate (i.e., 90%) as there is no human voice (e.g., lyric) in it. As for pop music, on account of the indistinct of the human voice (people won't even be able to hear lyrics clearly some times), it's also suitable as a carrier for malicious commands. But when it comes to the pure voice command generated by the text-to-speech engine, the attack success rate decreases largely. The pdf-likelihood is much higher in the posterior probability matrix computed from the pure voice command in comparison to popular music. There exists no noise and the DNN model is well trained, thus leads to a significantly higher probability of the correct pdf-ids compared with others, making it hard to make the malicious-pdf-ids have a higher possibility.

TABLE VII: The similarity between the original audio and the crafted one.

Audio Type	Sample Number	Iteration	Similarity
Pure Music	10	3000	97%
Pop Music	10	3000	96%
Voice Command	5	3000	94%

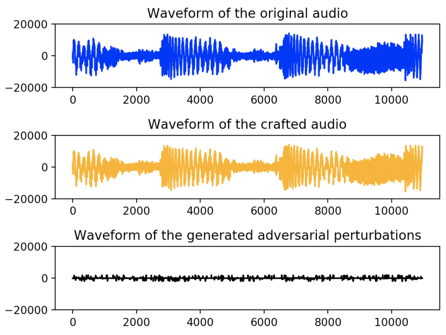

Fig. 2: Similarity between the original song and the crafted song (i.e., pure music).

C. *Similarity Between the Original Audio and the Crafted One*

To quantify the similarity between the original audio and the crafted one, we use the Pearson Correlation Coefficient (PCC) to calculate the similarity between the adversarial sample and the original audio clip. The correlation results are shown in Table VII. The similarity between the original audio input and the adversarial sample is relatively high, which is over 95%. This makes the added perturbation nearly impossible for humans to be noticed. We played adversarial samples to several volunteers, and none of them considered these adversarial samples are abnormal. From Figure 2, it's obvious that the original audio and adversarial sample only have little difference when using pure music as the audio input, and the perturbations added to it are relatively small. The similarity between them on average is 97%. Figure 3 is the waveform of the pure voice command. We can see that the similarity is still as high as 94%, though the perturbations are relatively more obvious.

VI. Conclusion

In this paper, we propose a semi-black-box adversary attack that can embed malicious voice commands into audio clips, and these embedded commands can be recognized by the ASR system Kaldi while remaining unnoticeable to humans. Different from existing studies, the proposed semi-black-box attack does not require the adversary to possess the full knowledge of the ASR system's DNN model. Using the genetic algorithm, we successfully construct adversary perturbations that are added into three kinds of audio clips. And the crafted audio clips (e.g., music or normal human command) can be successfully recognized as the inserted malicious commands by Kaldi, but remain a high similarity with the original audio, making them hard to be noticed for humans.

VII. Acknowledgment

This work was partially supported by the National Science Foundation Grants CNS-1514224, CNS-1820624, CCF-1801630, CCF-1909963 and Army Reasearch Office W911NF-18-1-0221.

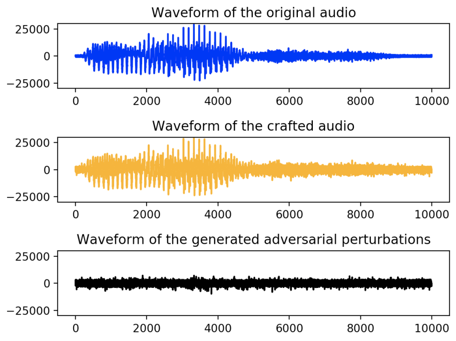

Fig. 3: Similarity between the original audio sample and the crafted audio (i.e., voice command).

References

[1] Text to speech online. https://ttsreader.com/.

[2] Moustafa Alzantot, Bharathan Balaji, and Mani Srivastava. Did you hear that? adversarial examples against automatic speech recognition. *arXiv preprint arXiv:1801.00554*, 2018.

[3] K. Audhkhasi, B. Kingsbury, B. Ramabhadran, G. Saon, and M. Picheny. Building competitive direct acoustics-to-word models for english conversational speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4759–4763, April 2018.

[4] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7. IEEE, 2018.

[5] Moustapha M Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6977–6987. Curran Associates, Inc., 2017.

[6] Lawrence Davis. Handbook of genetic algorithms. 1991.

[7] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[8] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[9] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[10] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The kaldi speech recognition toolkit. Technical report, IEEE Signal Processing Society, 2011.

[11] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. *arXiv preprint arXiv:1808.05665*, 2018.

[12] Rohan Taori, Amog Kamsetty, Brenton Chu, and Nikita Vemuri. Targeted adversarial examples for black box audio systems. *arXiv preprint arXiv:1805.07820*, 2018.

[13] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke. The microsoft 2017 conversational speech recognition system. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5934–5938, April 2018.

[14] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiaofeng Wang, and Carl A Gunter. Commandersong: A systematic approach for practical adversarial voice recognition. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 49–64, 2018.