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

Ganyu Wang
Ontario Tech University
Oshawa, Ontario
ganyu.wang@ontariotechu.ca

Miguel Vargas Martin
Ontario Tech University
Oshawa, Ontario
miguel.martin@ontariotechu.ca

2021 18th International Conference on Privacy, Security and Trust (PST) | 978-1-6654-0184-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/PST52912.2021.9647775

***Abstract*—Voice control systems continue becoming more pervasive as they are deployed in mobile phones, smart home devices, automobiles, etc. Commonly, voice control systems have high privileges on the device, such as making a call or placing an order. However, they are vulnerable to voice attacks, which may lead to serious consequences. In this paper, we propose SegmentPerturb which crafts hidden voice commands via inquiring the target models. The general idea of SegmentPerturb is that we separate the original command audio into multiple equal-length segments and apply maximum perturbation on each segment by probing the target speech recognition system. We show that our method is as efficient, and in some aspects outperforms other methods from previous works. We choose four popular speech recognition APIs and one mainstream smart home device to conduct the experiments. Results suggest that this algorithm can generate voice commands which can be recognized by the machine but are hard to understand by a human.**

***Index Terms*—Hidden Voice Attack, Automatic Speech Recognition, Smart Home Device.**

I. Introduction

Voice control systems are becoming increasingly popular and pervasive. For example, the well-known Siri from Apple, Cortana from Microsoft, iFLYTEK speech input, Baidu speech input. In 2020, the estimated population in the US who use a voice assistant at least monthly is 128 million, which represents 44.2% of internet users and 38.5% of the total population [5]. Particularly, for smart home devices such as Amazon Alexa, Google Home, etc. voice interface is the only interactive method for the users. Voice control is becoming one of the mainstream human-computer interaction modalities.

Attacking a voice assistant has a high return and a low risk for attackers. First, voice assistants always have a high priority in controlling the device, which means that compromising a voice assistant will give the attackers high-level control over the victim's devices. It is very common that a smart home device controls many electronic devices in the room. Furthermore, payment can be made by a voice assistant. Setting up payment with a voice assistant is even being advocated in the advertisement of the leading banks. Second, most of the voice control devices solely use conversation as the interactive method, the user cannot check what instructions the voice control devices have executed, which provides stealthiness for the attackers. Therefore, security should be considered an important issue for the design of voice assistants. A concrete example of how the attack works in a wide range is that the attacker embeds a malicious command in the audio of a famous YouTube video. The audio sounds similar to the original one but if it is played using the speaker, the voice control device nearby will recognize the malicious command and execute it.

However, the key technology of voice assistants, Automatic Speech Recognition (ASR), is highly vulnerable to a variety of attacks. Previous works have proven that the adversary can generate an audio sample that can be recognized as a meaningful command by the ASR, but at the same time being very hard for humans to understand (hidden voice attack) [7], [11], [26]. Another kind of voice attack is called adversarial attack, whereby the adversary tries to add the smallest amount of perturbation to the original audio, but making the ASR recognize it as another command which is totally different from the original command (adversarial attack on voice interface) [7], [12], [22], [28]. Other attacks take advantage of the property of the hardware, using ultrasonic waves to inject inaudible commands on the ASR [27], [29]. If the adversary deliberately designs the audio input, it is feasible for them to inject malicious commands on the voice control system.

This research studied the hidden voice attack on Automatic Speech Recognition (ASR), to help enhance the security of the voice control system. Previous research on hidden voice attacks is not practical enough for a real-world attack. The most stable attack (highest stealthy and success rate) on voice assistants is the hardware attack [24], [29], but those attacks normally need bulky instruments and close contact with the victim device. Hardware attacks work well for attacking a specific nearby target, but they do not work well for long-range and wide-range remote control attacks. Apart from hardware attacks, hidden voice attacks and adversarial attacks both use the speaker to attack the voice assistant, which means that they can be applied to long-range attacks as long as the adversaries compromise the speakers of the victims. However, to date, the research on voice attacks is still not practical in the real world, the attack distance is limited below 2 meters and only a few attacks [7], [11], [14], [28] work for the commercial ASRs. We design SegmentPerturb to perform a hidden voice attack on the ASR. Comparing with similar attacks, our attack produces equivalent or more unintelligibility in the attack audio samples and has a longer attack distance.

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

The contributions of our work are as follows.

- **SegmentPerturb.** A perturbation method to perform a hidden voice attack on any modern ASR system via inquiring is proposed. The idea of SegmentPerturb is to separate the audio into multiple equal-length segments and probe the ASR to distort the unimportant segments. Comparing with similar attacks, our attack produces equivalent or more unintelligibility in the attack audio samples and has a longer attack distance.
- **Full study on segmented perturbation (SegmentPerturb).** We present a full study on SegmentPerturb using four of the most popular speech recognition APIs in the market (Google, Wit, IBM, Azure). Besides, a real scenario attack using SegmentPerturb was demonstrated on Google Home. We set up a real scenario of the bedroom and placed the Google Home at different positions. Our attack samples can be recognized by the Google Home at most of the positions, which demonstrates the feasibility of this attack in the real world.
- **Our attack further relaxes the assumptions on the feature extraction method of the ASR for black-box attacks.** Previous works [7], [11], [26] on black-box hidden voice attack models have some assumptions on the signal processing phase of the ASR. But SegmentPerturb is based on the universal property which exists in all of the ASRs and even in human comprehension, that if the audio is separated into multiple equal-length segments, the importance of every segment is different. The ASR may pay more attention to some segments while somewhat ignoring others. Besides, because of the high noise resistance of modern ASRs, a large amount of perturbation can be added into almost every segment in one audio command.

II. Related Work

A. Software-level Voice Attacks

a) Hidden voice command: Hidden voice command is a kind of attack on ASR in which the adversary adds the largest perturbation on the audio, which totally destroys the intelligibility of the audio but the audio can still be recognized as a meaningful command by the ASR system. Carlini et al. [11] analyzed the hidden voice command in white-box and black-box settings. Under the white-box setting, they chose the CMU sphinx [19] (traditional speech recognition model) as the target model and generated hidden voice commands which cannot be understood by humans. Under a black-box setting, they can generate noise audio that can be recognized by machines but is hard to be understood by humans. Abdullah et al. [7] designed a practical hidden voice attack that attacked the signal processing phase of the ASR system. They added four types of perturbation (time domain inversion, random phase generation, high-frequency addition, time scaling) to the audio samples. Each kind of perturbation had some parameters and they generated a set of adversarial samples with different perturbation parameters and fed them into the ASR. They discarded the audio samples which cannot be transcribed correctly by the ASR and choose the audio file with the best parameter. Chen et al. [13] studied how to improve over-the-air voice attacks in a white-box setting, attacking the speech recognition model Deep Speech [16] from the Baidu Research team. They found that the reason for the difficulty of an over-the-air attack is the frequency-selectivity effect caused by the devices and channel, and they designed their over-the-air attack which achieved a 90% success rate over a distance of 6 m.

b) Adversarial attack against ASR: In an adversarial attack against the ASR system, the adversary adds the least possible amount of perturbation on the audio to make the machine recognize the audio as another transcription that is different from the original one. The adversarial attack on neural networks is first proposed by Szegedy et al. [25] in the computer vision domain.

Yuan et al. [28] performed the adversarial attack in a white-box setting and embedded the malicious command into the song. They performed their attack on the popular open-source speech recognition platform Kaldi [3], [21], which is a GMM-HMM speech recognition platform. They used gradient descent to craft the audio to approximate the same probability matrix of the audio of a meaningful command. Schönherr et al. [23] crafted adversarial attack samples in a white-box setting on the Kaldi platform [3] using a generative system based on neural networks. They were more inclined to add a component that is lower than the hearing threshold to prevent the human from perceiving the change. Qin et al. [22] further improved Schönherr et al.'s [23] work, using a similar psychoacoustic hiding method (using the hearing threshold to add perturbation) but attacked a modern Lingvo ASR system. They also made the adversarial samples more robust, having a better success probability for the over-the-air attack.

Kwon et al. [18] proposed a new kind of adversarial attack on ASR called selective audio adversarial samples. The selective audio adversarial sample is misclassified as the targeted phase by the victim ASR but is correctly classified by the protected ASR. Most of the previous adversarial attacks were based on white-box given the difficulty of perpetrating efficient black-box attacks. Chen et al. [14] proposed their so-called Devil's Whisper attack which used a known model plus a modern ASR to approximate the target black-box ASR model, and found that the adversarial samples for their self-built model have good transferability on the target model.

B. Hardware-level Voice Attack

There are other attacks using ultrasound or ultrasonic waves to perform inaudible voice attacks, which attack the signal processing hardware of a voice interface. Zhang et al. [29] took advantage of the non-linearity frequency response characteristic of the amplifier in the ASR and embedded malicious commands in an inaudible ultrasonic wave. Yan et al. [27] proposed an attack on the devices by transmitting malicious commands in ultrasonic waves through the solid medium. Sugawara et al. [24] proposed an attack that used laser light to control the voice control system, which was demonstrated to successfully attack various commercial products up to a 110-meter line of sight from the target device. While hardware-level voice attacks have much better concealment than software-level voice attacks, the hardware-level attacks need specifically designed devices to perform the attack, which limits the application scenario.

III. Method

A. Threat Model and Assumptions

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

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

B. Transmission Model

Two transmission models were used in this research, over-the-line (Section IV-B, IV-C) and over-the-air (Section IV-D).

a) Over-the-line: The over-the-line model is shown in Fig. 1 a). In the over-the-line setting, the adversarial audio is directly inputted to the target model as a file or stream data. There is no distortion from transmission in the over-the-line setting. The over-the-line attack was used in the feasibility test.

b) Over-the-air: The over-the-air model is shown in Fig. 1 b). Over-the-air is the most common and practical setting for voice attacks. In an over-the-air setting, the attacker would play the adversarial audio by an electronic speaker, and the target device would record this sound with an electronic speaker and input the recording to the target ASR model. The target model could be any model that can do speech recognition, such as cloud speech-to-text API or open-source speech-to-text model. It is also possible that the microphone and the target ASR model are in an integral device, such as Google Home, Amazon Alexa, etc.

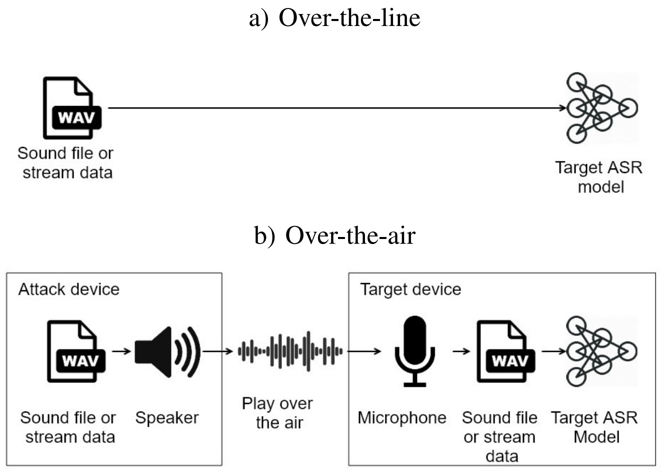


Fig. 1. Transmission Model.

C. Attack Scenario

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

The ASR system is regarded as a classifier $f_m(\cdot)$, while the input of the classifier is the waveform x of the sound signal recorded by the microphone, and the output of the ASR is a string of the transcribed result y_m. Similarly, human comprehension can also be regarded as a similar classifier $f_h(\cdot)$. The output y_h is the transcription of the waveform x from humans. Formally, this is:

$$\begin{cases} f_m(x) = y_m \\ f_h(x) = y_h \end{cases} \tag{1}$$

We define $S(x_0)$ as the set of the correct transcriptions of the original waveform x_0 for all of the target ASR. We get the correct transcription set of the waveform x_0 by inputting x_0 to all of the target ASR $f_i(\cdot)$ and add all transcriptions into set $S(x_0)$. We use Q to denote the index number list of all of the target ASR. The correct transcription set of x_0 is formally defined as:

$$S(x_0) = \{y | y = f_i(x_0), i \in Q\} \quad (2)$$

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

$$\begin{cases} f_m(x_0) \in S(x_0) \\ f_h(x_0) \in S(x_0) \end{cases} \quad (3)$$

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

$$\begin{cases} f_m(x^*) \in S(x_0) \\ f_h(x^*) \notin S(x_0) \end{cases} \quad (4)$$

D. Perturbation Framework

Basically, the attacker tries to reduce as much information as possible from the original waveform x_0 to get a perturbed waveform x^* which can still be recognized as the correct transcription by the machines ($f_m(x^*) \in S(x_0)$). Starting from $x^* = x_0$, we gradually increase the amount of perturbation on x^* to shift it to the classification margin f_m. If x^* has reached the classification margin, adding the smallest perturbation on x^* will make the machine generate an incorrect transcription $f_m(x^*) \notin S(x_0)$. Then we will take the x^* which is the closest to the margin and yield $f_m(x^*) \in S(x_0)$.

A schematic diagram is shown in Fig. 2. A small amount of perturbation was added into x_0 repeatedly, from x_1^* to x_2^* to x_n^*, until $f(x_{n+1}^*) \notin S(x_0)$. Then x_n^* will be accepted because $f(x_n^*) \in S(x_0)$ and it is the perturbed waveform that contains the least amount of information from the original waveform x_0.

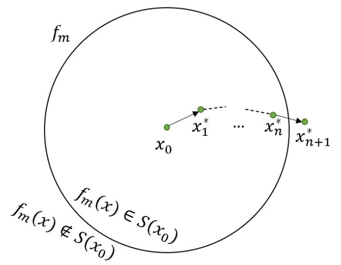


Fig. 2. Depiction of the iterative perturbation framework.

E. Monotonically Increasing Perturbation Function

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

The input of the perturbation function is the original waveform x and a perturbation parameter p. The output of the perturbation function is the perturbed waveform x^*.

MIPF should meet the following important properties.

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

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

Algorithm 1 MIPF: Random Delete

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

This simple function (Algorithm 1) is an instance of MIPF. It meets the three important properties mentioned above. First, if the perturbation rate p is set to 100% then every point in the perturbed waveform x^* is set to 0, thus no information of the original waveform x remains in x^*. Second, if the perturbation rate p is set to 0%, then the waveform x and the perturbed waveform x^* are the same because no point in x is deleted. Third, with the increase of p, more points in waveform x are deleted to get x^*, therefore lessening the information from x in x^*.

F. Naïve Perturbation

The general idea of the naïve perturbation consists of adding a small amount of perturbation δ to the whole audio waveform x^* at each loop using the MIPF. At each loop, we transcribed the perturbed waveform x^* to get the result $y^* = f_m(x^*)$. If $y^* \in S(x_0)$ then this small amount of perturbation does not make the x^* fall outside the classification margin, so we accept this perturbation. We do this repeatedly until $y^* \notin S(x_0)$. We would not accept the last δ and output the x^* which approximated the classification margin.

Using this basic idea, we proposed the naïve perturbation (Algorithm 2). The $perturbation\_rate\_list$ in Algorithm 2 is a list of the possible values within the domain of the definition of p, ordered from smallest to the largest. To find the marginal perturbation rate by traversing through the $perturbation\_rate\_list$ from the smallest to the largest is not efficient. Therefore, in our experiment, dichotomy was applied to efficiently get the highest perturbation rate for one given waveform x_0.

Algorithm 2 Naïve Perturbation

```
Input: $x, y_c, f$
Output: $x^*$
 1: for $p \in perturbation\_rate\_list$ do
 2:     $x_t^* \leftarrow \mathrm{MIPF}(x, p)$
 3:     $y \leftarrow f(x_t^*)$
 4:     if $y \in S(x)$ then
 5:         $x^* \leftarrow x_t^*$
 6:     else
 7:         break
 8:     end if
 9: end for
10: return $x^*$
```

G. Segmented Perturbation

To improve the perturbation degree, we propose a segmented perturbation algorithm (SegmentPerturb), which is an upgraded version of the naïve perturbation. It performs more stable than the naïve perturbation (according to the experiment in the next section) but uses more inquires to the model.

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

According to this theory, SegmentPerturb (Algorithm 3) was designed to get the highest perturbation rate for every segment. First, the audio waveform was separated into multiple equal-length segments. Then we started with the smallest perturbation rate, applying this perturbation rate to one segment in x_0, and then transcribing it using the ASR. If the ASR system outputs a correct transcription ($f_m(x^*) \in S(x_0)$), this small perturbation on this segment will be accepted; otherwise, this change will not be accepted (and use a flag "$meet\_end$" to mark that the perturbation rate for this segment has reached the perturbation boundary). The same procedures were repeated on all segments. Then the perturbation rate gradually increased until all of the segments had met the perturbation boundary or all of the perturbation rates had been traversed.

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

Algorithm 3 SegmentPerturb

```
Input: $x, y_c, f$
Output: $x^*$
 1: $x^* \leftarrow x$
 2: $perturb\_rate\_list$ is a list containing a sequence of ordered
    perturbation rates, for example [0.1, 0.2, 0.3, ... 0.9, 1.0].
 3: $segment\_list$ is the segments in $x$ which contain voice (random
    sequence).
 4: $meet\_end$ is the vector marking whether the perturbation rate
    of every segment has reached the perturbation boundary. It is
    initialized as all $False$.
 5: for $p \in perturb\_rate\_list$ do
 6:     for $b \in segment\_list$ do
 7:         if $meet\_end[b] == False$ then
 8:             $x_t^* \leftarrow x^*$
 9:             $x_b^* \leftarrow x_b$
                # Take out the segment $b$ for perturbation.
10:             $x_b^* \leftarrow \mathrm{MIPF}(x_b^*, p)$
                # Only apply the MIPF on the segment $b$
11:             replace the segment $b$ in $x_t^*$ with the $x_b^*$
12:             $y \leftarrow f(x_t^*)$
13:             if $y \in S(x)$ then
14:                 $x^* \leftarrow x_t^*$
                    # Accept this perturbation rate and submit the change.
15:             else
16:                 $meet\_end[b] \leftarrow True$
17:             end if
18:         end if
19:     end for
20: end for
21: return $x^*$
```

IV. Experiment

A. Experiment Setup

a) Hardware: The devices used in the experiment are ordinary devices that may be used domestically. A laptop that runs on Windows 10 was used for the experiment. The microphone in the experiment is an ARCHEER condenser microphone with a frequency response of 20 Hz to 20 kHz, a sensitivity of -34 dB ± 2 dB, and a signal-to-noise ratio of

78 dB. The electronic speaker in the experiment is EDIFIER R980T, which is an ordinary speaker with a signal-to-noise ratio of $\geq$ 85 dBA, distortion of $\leq$ 0.5%, and frequency response of 70 Hz to 20 kHz.

b) Target model: The target models were four commercial speech-to-text APIs plus one smart home device. The four commercial speech-to-text APIs were Google Speech Recognition [1], Microsoft Azure speech-to-text API [4], Wit API [6], and IBM speech services [2]. All of the speech recognition APIs mentioned above can be used through online requests. The smart home device used in the experiment was Google Home. The experiment is conducted on some of the most popular commercial speech recognition APIs and devices to verify the practicability of the attack in the real world.

c) Command selection and audio acquire: A representative command set of six commands are used in the experiment. The commands are “Turn on airplane mode”, “Open the door”, “Turn on the computer”, “Turn on the light”, “Call 911”, “Turn on Wi-Fi”. A similar command set was used in previous works [7], [11], [14] to represent a variety of the commands on the ASR system. We recorded our own voice commands in a quiet room.

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

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

B. Preliminary Experiment: Over-the-line Attack using Naïve Perturbation

The naïve perturbation under the over-the-line setting was tested as a preliminary experiment. We did this experiment for every command against every speech-to-text API. The perturbation rate p for every command list against all of the ASR is shown in Fig 3. For Google Speech API, most of the commands got a relatively high perturbation rate. “Turn on airplane mode”, “Call 911” and “Turn on Wi-Fi” even get around 0.9 perturbation rate, which means that even 90% of the points in waveform x are set to 0, the Google Speech API can still recognize it as the correct transcription. The naïve perturbation could only perturb as much as 70% and 15% on the Wit and the IBM speech recognition APIs respectively.

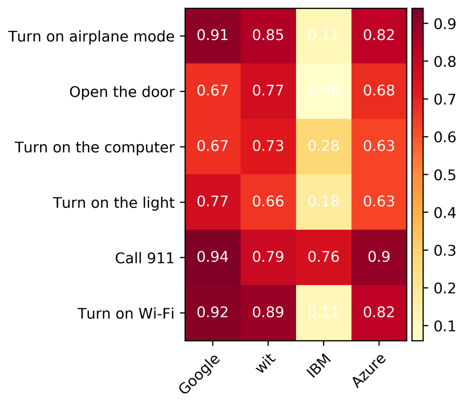


Fig. 3. Perturbation rate for the over-the-line attack using Naïve Perturbation.

C. Over-the-line Attack using SegmentPerturb

SegmentPerturb framework was tested using different metrics to represent the efficiency of the attack.

a) Heatmap: We use SegmentPerturb to attack the speech recognition APIs over-the-line. The segment length is originally set to 20 ms for this experiment, and the $perturb\_rate\_list$ is [0.2, 0.4, 0.6, 0.8, 0.9, 1.0].

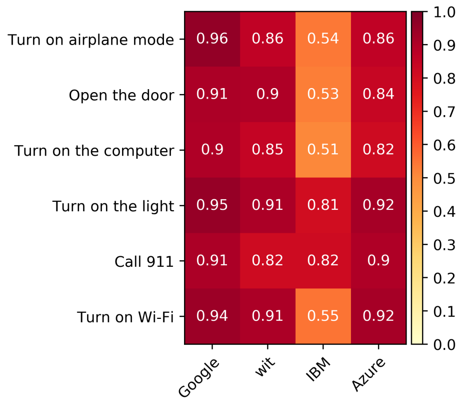
The Average Perturbation Rate (APR) was calculated for every command against each speech-to-text API (Fig. 4). APR is the average of the perturbation rate of all the segments that participated in the perturbation procedure.

As shown in Fig. 4 most of the APRs are higher than 0.8. Those audio samples are very difficult to understand by humans even if replayed multiple times (see Section IV-E), which indicates that this method is very successful in the over-the-line experiment.

Compared with Fig. 3 for Naïve Perturbation, the APR in Fig. 4 for all of the speech APIs showed steady improvement. Especially for the IBM speech services, in Fig. 3, the perturbation rate for most of the commands on IBM is lower than 0.3, which indicates that it is easy for humans to understand the perturbed voice command. But using SegmentPerturb, the APR exceeds 0.5 for all of the commands, some commands even got above 0.8.

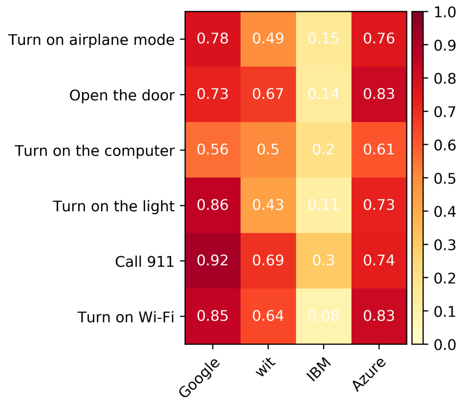
b) Segment length - APR: The relationship between segment length and APR was illustrated in Fig. 5. We generate the attack audio for every command twice against the Google speech recognition APIs, and use the APR to draw the boxplot figure. The segment length was set from 10 ms to 50 ms.

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



	Google	wit	IBM	Azure
Turn on airplane mode	0.96	0.86	0.54	0.86
Open the door	0.91	0.9	0.53	0.84
Turn on the computer	0.9	0.85	0.51	0.82
Turn on the light	0.95	0.91	0.81	0.92
Call 911	0.91	0.82	0.82	0.9
Turn on Wi-Fi	0.94	0.91	0.55	0.92

Fig. 4. The APR for over-the-line attack using SegmentPerturb - Random Delete.



	Google	wit	IBM	Azure
Turn on airplane mode	0.78	0.49	0.15	0.76
Open the door	0.73	0.67	0.14	0.83
Turn on the computer	0.56	0.5	0.2	0.61
Turn on the light	0.86	0.43	0.11	0.73
Call 911	0.92	0.69	0.3	0.74
Turn on Wi-Fi	0.85	0.64	0.08	0.83

Fig. 6. The APR for over-the-air attack using SegmentPerturb.

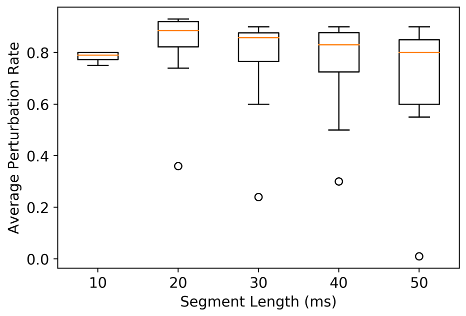


Fig. 5. The relationship between segment length and APR for "SegmentPerturb - Random Delete".

D. Over-the-air Attack using SegmentPerturb

a) Heatmap: We assessed SegmentPerturb in the over-the-air setting in this section. The parameters were the same as in Section IV-C; we use 20 ms time window and the $perturb\_rate\_list$ is [0.2, 0.4, 0.6, 0.8, 0.9, 1.0].

Fig. 6 shows the APR of the over-the-air attack experiment. In this experiment, the distance between the speaker and the microphone is one meter. For over-the-air attacks, the APR ranges from 0.50 to 0.92. IBM speech API can hardly recognize any original audio from over the air. Therefore, the perturbation rate tolerated by IBM was low. The amount of perturbation added into the audio command was highly related to the abnormal tolerance of the ASR. For each command, more perturbation can be added into the audio transcribed by Google, Wit, and Azure speech-to-text API.

b) Perturbation rate for every segment: The perturbation rate for every segment of the waveform was analyzed in this part. The first audio command "Turn on airplane mode" was randomly chosen to conduct this experiment. Results are shown in Fig. 7.

The waveform of the command was replayed in slow motion and we found that from 1.1s to 1.6s corresponds to "turn on", from 1.7s to 2.0s is for "airplane", and from 2.0s to 2.5s is for "mode". All of the segments which are the valid parts of the waveform took part in the perturbation procedure (from

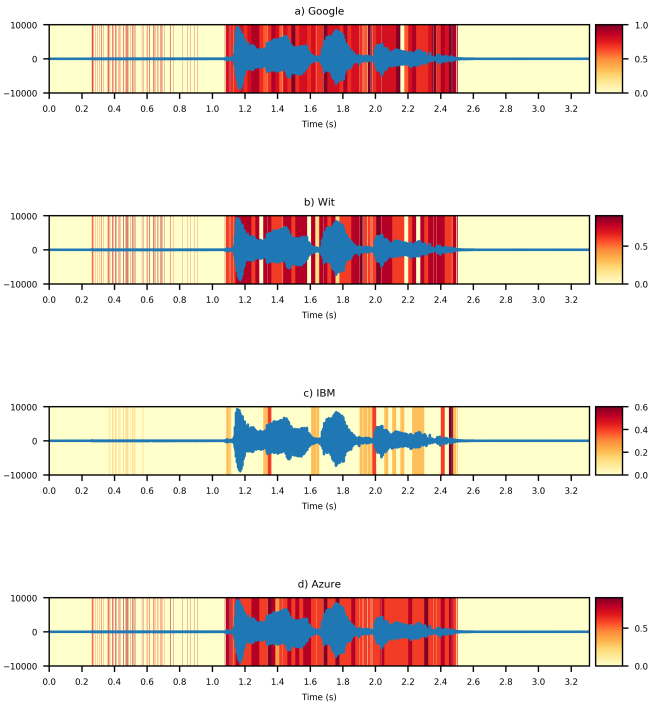


Fig. 7. The perturbation rate for every segment of the command "Turn on airplane mode".

1.1s to 2.5s). The segment whose amplitude is lower than the threshold did not participate in the perturbation procedure, therefore shown in light yellow (the perturbation rates for these segments are all 0). Some parts which should not be activated as the valid segments were activated because of the noise in the room (from 0s to 1s). But these segments only take a small proportion of all of the segments and will not make a difference in APR. In Fig. 7 Google, Wit, and Azure got high perturbation rates throughout the signal. The figure showed that the segments which are deemed important from the view of the machine are not the segments that are important from the view of the human. Humans pay more attention to the stress of the word (peak of the signal), but those parts can be added more perturbation with SegmentPerturb.

c) Attack distance - APR: The performance of the attack in different ranges of distances was tested in this part. We used the Google speech recognition API for this test. The distance from the speaker to the microphone was the variable of the experiment and the APR for generating the audio command samples was observed. The result for the distance and APR is illustrated in Fig. 8. When the distance is one meter, the mean APR is around 0.8. From 1.5 m to 2.0 m the mean APR remains above 0.7. As the distance increases to 2.5 meters, the mean APR remains at 0.6. According to the audio intelligibility in the next section, if the APR is higher than 0.7, the probability for a human to comprehend the meaning of the command could be very low. Therefore, two meters seems to be an effective distance for our attack.

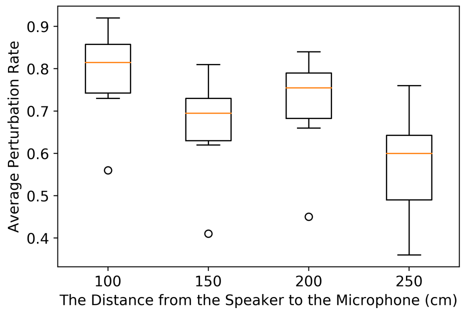


Fig. 8. The relationship between the physical distance and the APR.

d) Attack on Google Home: We demonstrated the feasibility of our attack on Google Home. Because we cannot get the transcription from Google Home device, we set up the smart plugs and give the smart plugs different names ("airplane mode", "computer", "light", "Wi-Fi") to test if the hidden voice command was correctly transcribed. If the command is correctly recognized by Google Home, the smart plugs would be switched on.

We used the attack samples[1] obtained from attacking Google speech-to-text API over-the-air. Only the "turn on [device]" commands were tested, because if the smart plug was turned on, we can be sure that the command is correctly executed by Google Home. For other commands, Google Home will reply "Sorry, I cannot understand" for "Open the door". Besides, we do not want to make accidental calls to the 911 emergency services for testing the "Call 911". Therefore, "Open the door" and "Call 911" were excluded, and we only tested the four "Turn on [device]" commands.

To test the feasibility we perform a real scenario attack in an actual bedroom. The bedroom's floor plan is depicted in Fig. 9. Google Home was placed at positions A to F. The speaker was placed at position S. The distance from the speaker S to each position was S-A (50 cm), S-B (220 cm), S-C (150 cm), S-D (270 cm), S-E (270 cm), and S-F (370 cm). All positions were at the same horizontal plane which is 75 cm above the ground, common height for an office desk. We set the system volume to 60 and see if the attack audio can be recognized by Google Home. The result is shown in Table I. Most commands can be recognized by Google Home at each position. At position A, the command was too loud that one of the commands was not recognized by Google Home. To our surprise, at position F, the farthest position, all commands were correctly recognized by Google Home. Therefore, a real-world attack using SegmentPerturb could be feasible.

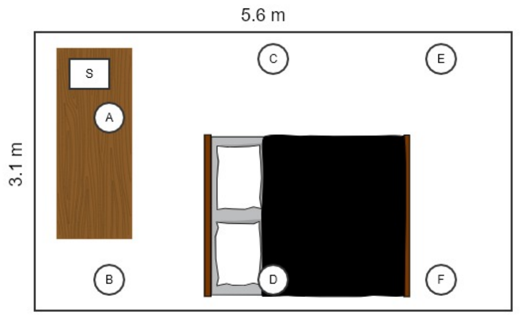


Fig. 9. A floor plan of the bedroom. Google Home was placed at positions A through F.

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

TABLE I
REAL SCENARIO ATTACK ON GOOGLE HOME. ✓: THE COMMAND CAN BE RECOGNIZED BY GOOGLE HOME. ✗: THE COMMAND CANNOT BE RECOGNIZED BY GOOGLE HOME.

E. Audio Intelligibility Study

We conducted the intelligibility experiment[2] in the form of a questionnaire in Amazon Mechanical Turk to evaluate the intelligibility of the audio commands we designed. The goal of the experiment is to test the intelligibility of the normal audio clips and the SegmentPerturb audio clips. The audio intelligibility test uses the attack audio samples[3] obtained from attacking Google speech-to-text API over-the-air.

200 participants were recruited to participate in the intelligibility study. The participants were assigned two groups (pool A and pool B), with 100 participants per group. Participants

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

[2]The experiment was approved by our institution's Research Ethics Board [16091].

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

Command (APR)	Normal		Obfuscated	
	1st	2nd	1st	2nd
Turn on airplane mode (0.78)	63%	76%	1%	2%
Open the door (0.73)	96%	100%	27%	38%
Turn on the computer (0.56)	92%	96%	42%	57%
Turn on the light (0.86)	93%	96%	20%	32%
Call 911 (0.92)	93%	93%	15%	22%
Turn on Wi-Fi (0.85)	96%	100%	45%	65%

TABLE II
AUDIO INTELLIGIBILITY TEST.

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

Participants were asked to play each command twice and input what they were able to understand each time. The whole survey lasted for about 4 minutes. Participants were compensated with an amount prorated according to the minimum wage in the area where our research facility is located. The results exclude the data from 6 participants who did not get a correct answer for the validation audio or reported a technical issue (there is a question at the end of the questionnaire to find out if the participant encountered technical issues).

We counted how many of the participants inputted the correct transcription and divided it by the number of validated data in that pool to get the percentage of the participants who recognized the command. These percentages represent the intelligibility of the command. Results are shown in Table II. All of the normal commands got a high score in intelligibility, which indicates that those commands are clear, while obfuscated commands got much lower in intelligibility scores. This indicates that SegmentPerturb has a good ability to reduce intelligibility. In this experiment, the participants are told to pay attention to the audio and try to recognize it. If this attack happens in the real world, the user of the victim device will not be alerted to pay attention to the audio, thus few users would notice the attack.

V. DISCUSSION

A. Compare with Other Over-the-air Black-box Voice Attacks

Following the comparison table proposed by Abdullah et al. [8], we compare our work with other works that use a similar attack scenario. We selected the black-box voice attack that works over-the-air using speakers and microphones, excluded the attacks using special devices and mediums, such as ultrasound or laser light. The selected works are Devil's Whisper [14], Abdullah et al.'s [7], Cocaine Noodles [26], Hidden voice command [11]. The comparison is shown in Table III.[4]

For audio type, Devil's Whisper [14] is an adversarial attack that provides much more stealthiness than all other attacks, but to generate an attack model needs 4.6 hours of training. Our attack use around 300 queries for generating one attack audio sample, which could be better used in the scenario that the attacker only needs a few attack samples. Compared with Abdullah's attack [7], although our attack consumes more computational power, we achieve a further attack distance which makes the attack much more realistic. Especially, according to the real scenario experiment, our attack has the potential to perform an attack at a distance of more than 2 meters in an indoor environment.

B. Detailed Comparison with State-of-the-art Hidden Voice Attacks

The attack from Abdullah et al.'s [7] is reproduced in this section according to the code provided by the authors.[5]

The authors provided the code of their four attacks, which are time domain inversion (TDI), random phase generation (RPG), high frequency addition (HFA), and time scalling (TS). However, they did not provide the code of their so-called "perturbation engine" they refer to in [7], which is meant to combine the four attack methods to generate the "best parameter set" for their attack. In lieu of the perturbation engine, the evaluation was done on each attack method they proposed, following the conclusion provided in [7] and choosing the parameter which best obfuscates the audio while it can still be recognized by ASR. The attack samples[6] represented the actual attack efficiency.

A similar intelligibility test was conducted on the audio sample reproduced according to Abdullah et al.'s work [7]. We only tested TDI and RPG because only these two methods obfuscated the audio, while the other two either consisted of adding high-frequency sine waves (HFA) or doing a tempo change (TS) which do not make the audio obfuscated. TDI and RPG can be regarded as the essential methods in their attack. Therefore, testing on RPG and TDI can represent their attack efficiency to a great extent.

The experiment procedure follows the same principle as in section IV-E. The ratio of the participants who recognized the commands is shown in Table IV. According to the table, the average for first-time recognition of the TDI samples is 73.5%, and that for RPG is 80.3%, which is higher than ours 25%. Therefore, compared with theirs, our attack did better in reducing intelligibility.

[4]Medium: "L", over-the-line; "A", over-the-air. # Queries & Time : "N/A", the authors did not provide that information. ASR internal: "BB", the attack can be used in a black-box setting.

[5]The link provided in [7] is https://github.com/hamzayacoob/VPSesAttacks

[6]The attack samples are available to listen in this website https://ganyuwang.github.io/Compare_Practical.html.

Attack Method	Audio Type	Medium	# Queries	Time	# ASR Attacked	ASR internals	Attack distance	Transferability
Devil's Whisper [14]	Clean	L/A	1500	4.6h	3	BB	0.05-2 m	Yes
Abdullah et al. [7]	Noisy	L/A	10	Seconds	12	RNN,HMM,BB	0.3 m	Yes
Cocaine Noodles [26]	Noisy	L/A	N/A	N/A	1	BB	0.3 m	No
Hidden voice command [11]	Noisy	L/A	N/A	32h	1	RNN	0.5 m	No
SegmentPerturb	Noisy	L/A	Around 300	30min	4	BB	0.5-2 m	No

TABLE III
COMPARING WITH OTHER OVER-THE-AIR BLACK-BOX VOICE ATTACKS.

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

TABLE IV
AUDIO INTELLIGIBILITY TEST FOR [7]

C. SegmentPerturb as a Framework

SegmentPerturb is a framework for crafting hidden voice commands by inquiring the ASR. We start with the ordinary voice command, separate the audio into equal-length time windows, and apply perturbation on each segment while probing the ASR to judge if the perturbation is valid.

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

D. Judging Whether the Voice Attack Truly Fools the ASR

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

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

VI. LIMITATIONS

A. Limited Transferability

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

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

B. Attack Performance Depends on Abnormal Tolerance of ASR

The success rate of our attack is highly related to the abnormal tolerance of the ASR model. If the model is more noise-tolerant, more segments in the waveform can be perturbed to a higher degree. In Fig. 3 and Fig. 4, Google and Azure are more noise-tolerant as they can recognize a meaningful command from strongly perturbed audio. For IBM speech recognition, even a small perturbation rate on the audio will make it generate the wrong output. The noise-resistant property of the ASR needs to be carefully designed for preventing this kind of attack. If the ASR system is more robust, it would be easier for the attacker to perform a hidden voice attack with a more unintelligible voice command.

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

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

VII. Defence

A. Electronic Sound Detection

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

To prevent voice attacks, the "ASVspoof Challenge" began in 2015. In ASVspoof Lavrentyeva et al. [20] used Deep Neural Network to distinguish the replay spoof attack and they achieved an equal error rate (EER) of 6.73% which had a 72% relative improvement over the baseline system from ASVspoof 2017. Blue et al. [10] studied the difference between electronic and human voice. They studied the frequency domain characteristics at the sub-bass (20-80Hz) region and used this characteristic to distinguish electronic speakers and the human voice. Their classification result gets a high true positive rate while keeping the false positive rate at a minimum. Ahmed et al. [9] further improved the method of detecting electronic sound; they proposed a fast and light electronic sound detection model, which only takes in 97 features. However, the defence method of detecting electronic sound may fail if the adversary carefully designs the adversarial audio or uses a high-quality audio device.

B. Audio Reconstruction

In an adversarial attack, a small perturbation δ is added to the original audio. Therefore, it is normally hard for the small and delicate perturbation δ to survive a reconstruction of the audio. For example, Eisenhofer et al. [15] proposed an audio transform in the ASR pipeline to resemble human hearing. Applying the transform, the adversarial example will either be filtered out or can be easily perceived by a human. Hussain et al. [17] designed an adversarial example detector, using the idea of reconstructing the audio. The idea is that after applying some reconstruction techniques (for example, down-sampling and up-sampling) the transcriptions for benign audio should not change too much from the original. If the transcription of the audio changes a lot after the audio reconstruction, this audio is deemed as an adversarial example. Using this idea their detector achieved more than 90% of accuracy in detecting the latest adversarial attacks. It is very likely that our attack is filtered out with the audio reconstruction-based defence method.

VIII. Conclusion

We proposed a novel attack method for the black-box hidden voice attack on modern ASR systems. The goal of the attack is to generate noise-like attack audio samples that can be recognized by the ASR but cannot be recognized by the human. SegmentPerturb is tested using four commercial speech-to-text APIs plus one mainstream smart home device. This research alerts the industry and consumers of voice control devices, that the security of ASR deserves more attention, because only by using a probe and perturb framework, the attack can generate efficient hidden voice attack samples, and these samples could be broadcast to a large scale.

In over-the-line attacks, SegmentPerturb can generate audio commands which can be recognized by commercial ASRs but cannot be recognized by a human. In over-the-air attacks, a series of practical experiments were performed. In a real scenario attack, we place the Google Home at six representative positions in the bedroom, and most of the commands can be recognized at each position. The audio intelligibility test showed that only around 25% of participants can recognize the command when they first hear it. Therefore, the attack audio samples have some practicability and stealthiness in the real world.

Acknowledgement

We thank the support of the Natural Sciences and Engineering Research Council of Canada, Grant # RGPIN/005919-2018.

References

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

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

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

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

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

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

[7] Hadi Abdullah, Washington Garcia, Christian Peeters, Patrick Traynor, Kevin R. B. Butler, and Joseph Wilson. Practical hidden voice attacks against speech and speaker recognition systems. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

[8] Hadi Abdullah, Kevin Warren, Vincent Bindschaedler, Nicolas Papernot, and Patrick Traynor. SoK: The Faults in our ASRs: An Overview of Attacks against Automatic Speech Recognition and Speaker Identification Systems. *arXiv e-prints*, pages arXiv–2007, 2020.

[9] Muhammad Ejaz Ahmed, Il-Youp Kwak, Jun Ho Huh, Iljoo Kim, Taekkyung Oh, and Hyoungshick Kim. Void: A fast and light voice liveness detection system. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2685–2702, 2020.

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

[11] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David A. Wagner, and Wenchao Zhou. Hidden voice commands. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pages 513–530. USENIX Association, 2016.

[12] 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.

[13] Tao Chen, Longfei Shangguan, Zhenjiang Li, and Kyle Jamieson. Metamorph: Injecting inaudible commands into over-the-air voice controlled systems. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.

[14] Yuxuan Chen, Xuejing Yuan, Jiangshan Zhang, Yue Zhao, Shengzhi Zhang, Kai Chen, and XiaoFeng Wang. Devil's whisper: A general approach for physical adversarial attacks against commercial blackbox speech recognition devices. In *29th USENIX Security Symposium (USENIX Security 20)*, 2020.

[15] Thorsten Eisenhofer, Lea Schönherr, Joel Frank, Lars Speckemeier, Dorothea Kolossa, and Thorsten Holz. Dompteur: Taming audio adversarial examples. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2309–2326. USENIX Association, August 2021.

[16] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014.

[17] Shehzeen Hussain, Paarth Neekhara, Shlomo Dubnov, Julian McAuley, and Farinaz Koushanfar. WaveGuard: Understanding and mitigating audio adversarial examples. In *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

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

[19] Paul Lamere, Philip Kwok, Evandro Gouvea, Bhiksha Raj, Rita Singh, William Walker, Manfred Warmuth, and Peter Wolf. The CMU sphinx-4 speech recognition system. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2003), Hong Kong*, volume 1, pages 2–5, 2003.

[20] Galina Lavrentyeva, Sergey Novoselov, Egor Malykh, Alexander Kozlov, Oleg Kudashev, and Vadim Shchemelinin. Audio replay attack detection with deep learning frameworks. In *Interspeech*, pages 82–86, 2017.

[21] 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. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, number CONF. IEEE Signal Processing Society, 2011.

[22] Yao Qin, Nicholas Carlini, Garrison Cottrell, Ian Goodfellow, and Colin Raffel. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5231–5240, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[23] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

[24] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. Light commands: laser-based audio injection attacks on voice-controllable systems. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2631–2648, 2020.

[25] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[26] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. Cocaine noodles: exploiting the gap between human and machine speech recognition. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*, 2015.

[27] Qiben Yan, Kehai Liu, Qin Zhou, Hanqing Guo, and Ning Zhang. Surfingattack: Interactive hidden attack on voice assistants using ultrasonic guided waves. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.

[28] 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 William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 49–64. USENIX Association, 2018.

[29] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 103–117, 2017.

Appendix A
Audio Intelligibility Test

A. Instructions

a) What you need to do: This experiment tests the intelligibility of up to 10 audio clips with noise. You will need to play each audio clip twice.

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

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

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

Repeat the same procedure for all audio clips.

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

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