Your Voice is My Passport

John Seymour and Azeem Aqil

Abstract

Financial institutions, home automation products, and hi-tech offices have increasingly used voice fingerprinting as a method for authentication. Recent advances in machine learning have shown that text-to-speech systems can generate synthetic, high-quality audio of subjects using audio recordings of their speech. Are current techniques for audio generation enough to spoof voice authentication algorithms? We demonstrate, using freely available machine learning models and limited budget, that standard speaker recognition and voice authentication systems are indeed fooled by targeted text-to-speech attacks. We further show a method which reduces data required to perform such an attack, demonstrating that more people are at risk for voice impersonation than previously thought.

Contents

Introduction

Gaming voice systems is not a new idea[1]. Yet, voice authentication methods have been deployed in financial institutions to reduce fraud[2], home automation systems to allow for simpler shopping[3], mobile phones for command recognition[4], to name a few. Voice systems are generally susceptible to replay attacks, though some systems avoid this by forcing a would-be attacker to generate new, unknown, or random audio as part of the authentication mechanism[5]. The notion of voice as a biometric is so ingrained in human culture that people and courts are prone to trust audio recordings as evidence.

What *is* new is that the quality of synthetic speech generators have approached that of professional recordings[6] using only recorded speech from the target, along with other things that can be derived from it (e.g. transcripts). The field is so full of research opportunity that there are workshops at top tier machine learning conferences devoted to the subject[7]. Although such a technology is well-intentioned, there are methods for abuse. First, audio of a political candidate can easily influence public opinion, and demonstrations for generating such audio already exist[8] as well as generation of accompanying video[9]. Second, attackers may be able to unlock mobile devices or issue malicious commands[10]. Third, attackers can impersonate executives and bypass voice fingerprinting to authorize extremely large financial transactions (perhaps by chaining an otherwise expensive SS7 exploit[11] to forward that executive's calls to the attacker).

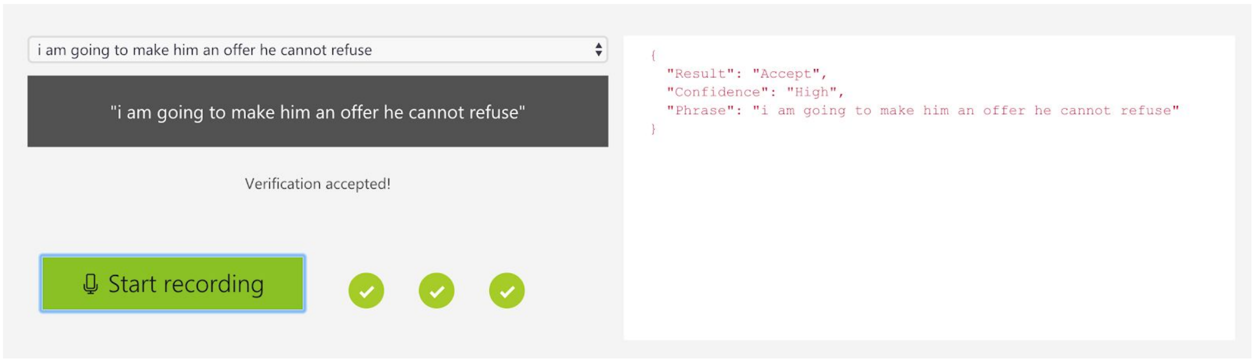
Most prior work attacking such systems have utilized Generative Adversarial Networks (GANs), training the attacker model against a given voice authentication system[12]. While this method has some desirable properties, for example the possibility of hiding commands within benign-sounding audio, the method is currently brittle: state-of-the-art techniques for hidden voice commands introduce corruption when played over-the-air. We use the simpler approach of generating speech for a given user, accepting the potential issues with data quantity: most text-to-speech systems require on the order of 24 hours of speech in order to train. Luckily, it seems the more valuable the target, the more likely audio of the target exists from political speeches, conference keynotes, or press releases.

[1] https://www.youtube.com/watch?v=-zVgWpVXb64
[2] http://www.newsweek.com/science-voice-recognition-fingerprints-barcleys-511561
[3] https://www.amazon.com/gp/help/customer/display.html?nodeId=202199440
[4] https://www.apple.com/ios/siri/
[5] https://azure.microsoft.com/en-us/services/cognitive-services/speaker-recognition/
[6] https://research.googleblog.com/2017/12/tacotron-2-generating-human-like-speech.html
[7] https://sites.google.com/site/nips2016endtoendspeechaudio/
[8] https://lyrebird.ai/demo/
[9] https://www.youtube.com/watch?v=9Yq67CjDqvw
[10] https://nicholas.carlini.com/code/audio_adversarial_examples/
[11] https://berlin.ccc.de/~tobias/31c3-ss7-locate-track-manipulate.pdf
[12] https://arxiv.org/pdf/1801.01944.pdf

Proof of Concept

We first demonstrate the method works generally using Lyrebird[13]: a service allowing users to generate audio from their own voice. Users generate speech using Lyrebird by creating an account, recording themselves reciting a minimum of 30 sentences, and providing a target sentence that Lyrebird will generate. The 30 sentences are chosen by Lyrebird, and are the same for all users.

There is some degradation of quality in generated audio, so it's not immediately obvious whether or not this audio will be accepted by common speaker recognition software. In order to test the idea, we set up Apple Siri and Microsoft Speaker Recognition API to recognize our voices, generate the target passphrases ("Hey Siri" and "I am going to make him an offer he cannot refuse") using Lyrebird, and test the audio against the speaker recognition software. Both services authenticated with the generated audio.


i am going to make him an offer he cannot refuse
"i am going to make him an offer he cannot refuse"
Verification accepted!
Start recording

```
{
  "Result": "Accept",
  "Confidence": "High",
  "Phrase": "i am going to make him an offer he cannot refuse"
}
```

Figure 1. Generated audio is accepted by Microsoft Speaker Recognition API.

This proof of concept is of limited value from a security standpoint: Lyrebird requires specific utterances in order to generate audio, and it is unlikely that an attacker could obtain such recordings from a target (though this does mean that the Lyrebird database, as well as voice authentication databases in general, could be valuable targets for other attacks). To demonstrate how a real attack might work, we turn to the state-of-the-art in text-to-speech generation.

[13] https://lyrebird.ai/

Text-to-Speech Methods

Text to speech (TTS) is generally considered to be a particularly difficult machine learning problem. Consider for example the myriad permutations of speaking styles, emotions, accents and pronunciations that can all map to the same text. However, recent (in the past year) advances in machine learning methods have drastically increased the quality of generated speech- to the point of being indistinguishable from the actual speaker[14].

Most of these systems are trained on ~24 hours of transcribed audio, broken down into short phrases or sentences. Mel-spectrograms are the predominant method for featurizing the audio: frames of audio are converted into numerical vectors which can be consumed by models. And finally, most of these systems utilize the hallucinatory abilities of deep learning systems in order to generate speech. Beyond these characteristics, however, TTS methods vary wildly depending on implementation.

We focus on two general models for TTS: Tacotron and Wavenet (though there are many variations even of these and many other options). Audio samples for Tacotron v1, Tacotron v2, and WaveNet are available[15,16,17]. To generalize our threat model as much as possible, we don't wish to assume any knowledge of nor make any modifications to models, we want to have a black box that we shove audio data and transcripts into and obtain reasonable-quality speech. The best open-source versions we can find for these families of models are available on Github [18,19], though Tacotron v2 isn't currently implemented and open-source implementations currently suffer from a degradation in audio quality[20,21]. However, this is likely a temporary limitation.

There are two leading datasets for speech generation: LJ Speech[22], made up of 13,100 audio clips with transcripts, and Blizzard[23], made up of 16k audio clips with transcripts. Both of these datasets contain on the order of 24 hours of speech data. Recent papers[24] also use Obama speeches to train, as audio and transcripts are available online[25] (though not broken up into sentences).

[14] https://research.googleblog.com/2017/12/tacotron-2-generating-human-like-speech.html
[15] https://google.github.io/tacotron/publications/tacotron/index.html
[16] https://google.github.io/tacotron/publications/tacotron2/index.html
[17] https://deepmind.com/blog/wavenet-generative-model-raw-audio/
[18] https://github.com/keithito/tacotron
[19] https://github.com/r9y9/wavenet_vocoder
[20] https://keithito.github.io/audio-samples/
[21] https://r9y9.github.io/wavenet_vocoder/
[22] https://keithito.com/LJ-Speech-Dataset/
[23] http://www.cstr.ed.ac.uk/projects/blizzard/2011/lessac_blizzard2011/
[24] http://grail.cs.washington.edu/projects/AudioToObama/siggraph17_obama.pdf
[25] http://www.americanrhetoric.com/

Data, Augmentation, and Transference

After performing some initial tests with the standard datasets, we turn our focus to generating audio for a given target. We first attempt to simply download audio from e.g. Youtube, and use automated services like Google's Cloud Speech API to transcribe it. This doesn't work well, because 1) the transcripts remove "mistake" words like "um", and 2) the api is not perfect, and we worry that any mistakes in the transcripts will influence the models.

Because transcribing audio takes a lot of effort, we return to our initial assumptions about the limitations of TTS methods (namely those surrounding the amount of speech data required). We attempt two methods to quantify the amount of data required for training such a system: naively training the model on limited data, and transfer learning. We found limited success in training a model directly on small amounts of audio from one of our previous DEF CON talks[26]: the audio quality is relatively low, amount of data is strongly correlated with how well the model trains, and the speaker makes an abnormally large number of mistakes. However, transfer learning seems to work well- if one trains a model on LJ Speech dataset, then swaps out to the new, smaller dataset, output from the model is understandable and sounds more like the new dataset.

This works because deep learning is centered on the idea that lower layers learn abstract concepts, for example, an image recognition system might have a layer which learns edges and lines, which feeds into a layer which learns shapes, which eventually feeds into the final layer which learns whether an object is a dog or a cat. We believe there might be a similar effect in TTS systems: the lower layers learn how to match text to words, and the higher layers learn things like pitch and accent. In order to transfer to a new dataset, the model doesn't need to change base beliefs, such as how text maps to words (which incurs a large training cost).

However, only using transfer learning is not enough to produce quality audio- we still need hours of transcribed audio to perform well. We also drastically augment our training data by changing the speed and pitch of the audio slightly (up to a tenth of an octave down or up). For each audio recording, we produce 20 new recordings in this way. This converts 5 minutes of audio into almost 2 hours- which is now easy to collect/transcribe. This way, we can focus on getting high-quality recordings, rather than spending all of our effort on transcribing.

Mitigations, Discussion, and Conclusions

The strategies employed above reduce the amount of time and effort for attacking speech authentication methods to downloading some open source code, reading a few StackOverflow posts, spending a few hours transcribing audio, and running code on a decent GPU for a little over a weekend. Our work makes it clear that speech authentication, by itself, is vulnerable,

[26] https://www.youtube.com/watch?v=w8VNY9k8i6o

when the victim has publicly available recorded audio, and the speech prompt is known to the attacker. Speaker recognition is **not** true 2FA.

By far, the biggest hurdle to a successful TTS attack is data quality. Audio quality, for instance, background noise or fuzzy voice, has a huge impact on the quality of output speech. Transcripts must be perfect as well; we have not had success with automated transcription services because of small errors, the postprocess removal of words like “Uh”, and the fact that sometimes only a part of a word is uttered. We’ve had to resort to removing pieces of audio which are not high-quality as a preprocessing step, which takes a lot of effort and increases the amount of audio necessary.

We know of two methods which would successfully mitigate this attack. If the speech prompt is not known to the attacker and there is a large dictionary of potential prompts (i.e. a passphrase), then voice authentication is at least as strong as the passphrase. We understand there are issues with accuracy in speaker recognition when the phrase can vary from individual to individual, but instead of relying on speaker recognition, this method relies on the passphrase itself to authenticate, and speech is only used as a means to transmit the passphrase. The second method is adding a component which distinguishes between human and computer generated speech; however, this method would still be vulnerable to GANs and adversarial attacks. We strongly prefer variations on the first method.

Given the ease with which voice authentication can be subverted, we recommend not using voice as a means to authenticate until a viable solution to one of the root enablers of this attack is found. As of today, we know of no mechanism that can do this reliably.