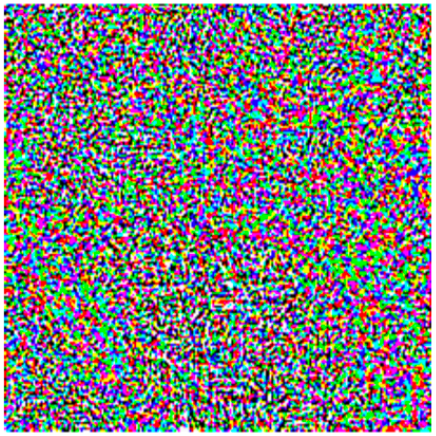
A Multiversion Programming Inspired Approach to Detecting Audio Adversarial Examples

Qiang Zeng,
Jianhai Su, Chenglong Fu, Golam Kayas, Lannan Luo,
Xiaojiang Du, Chiu C. Tan, and Jie Wu

DSN 2019

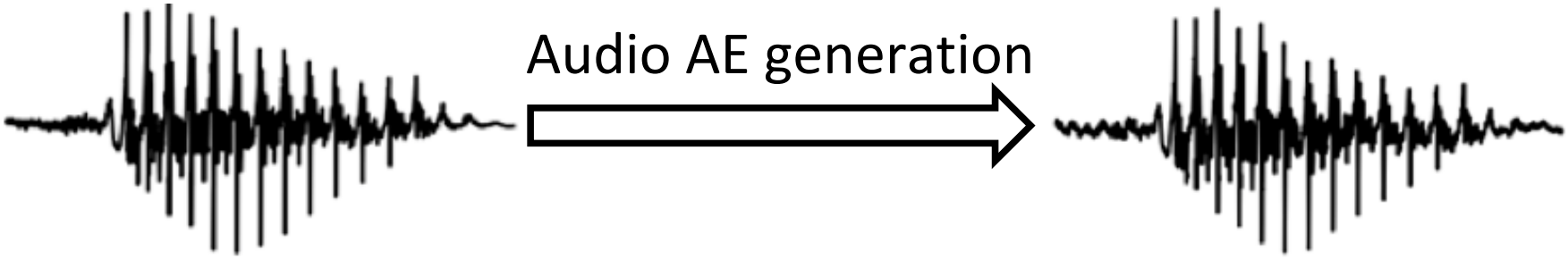
$+ .007 \times$



$=$

“panda”
57.7% confidence

“gibbon”
99.3 % confidence



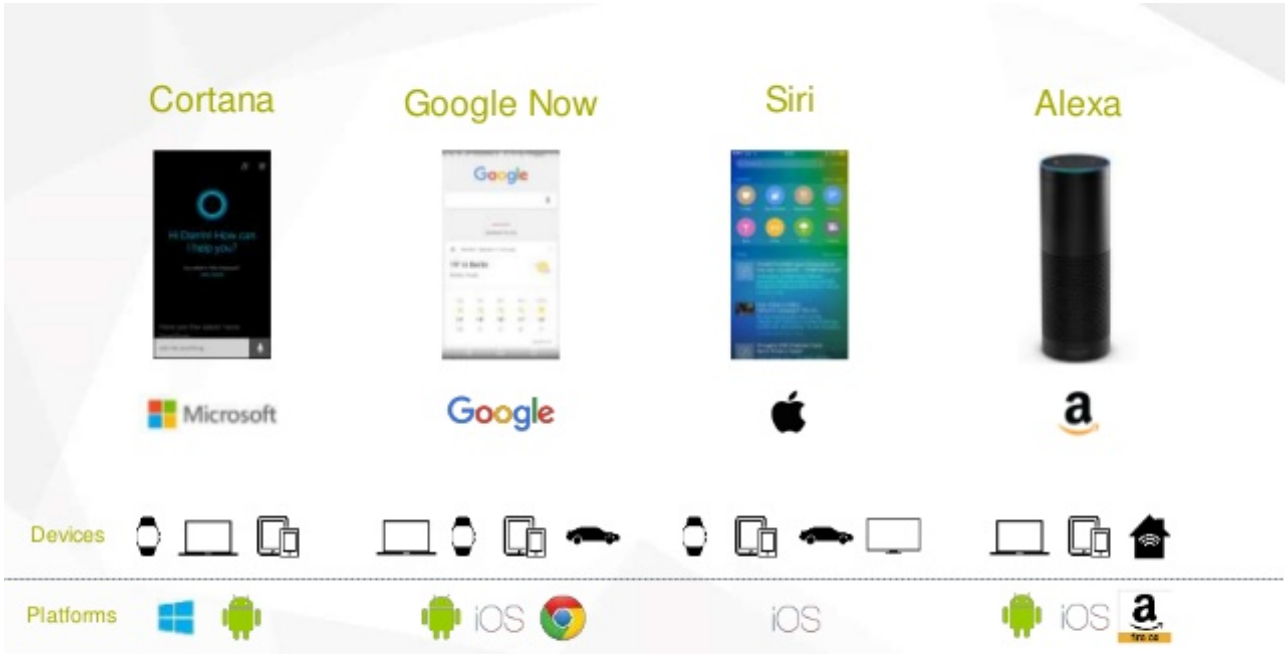
“I wish you wouldn’t”

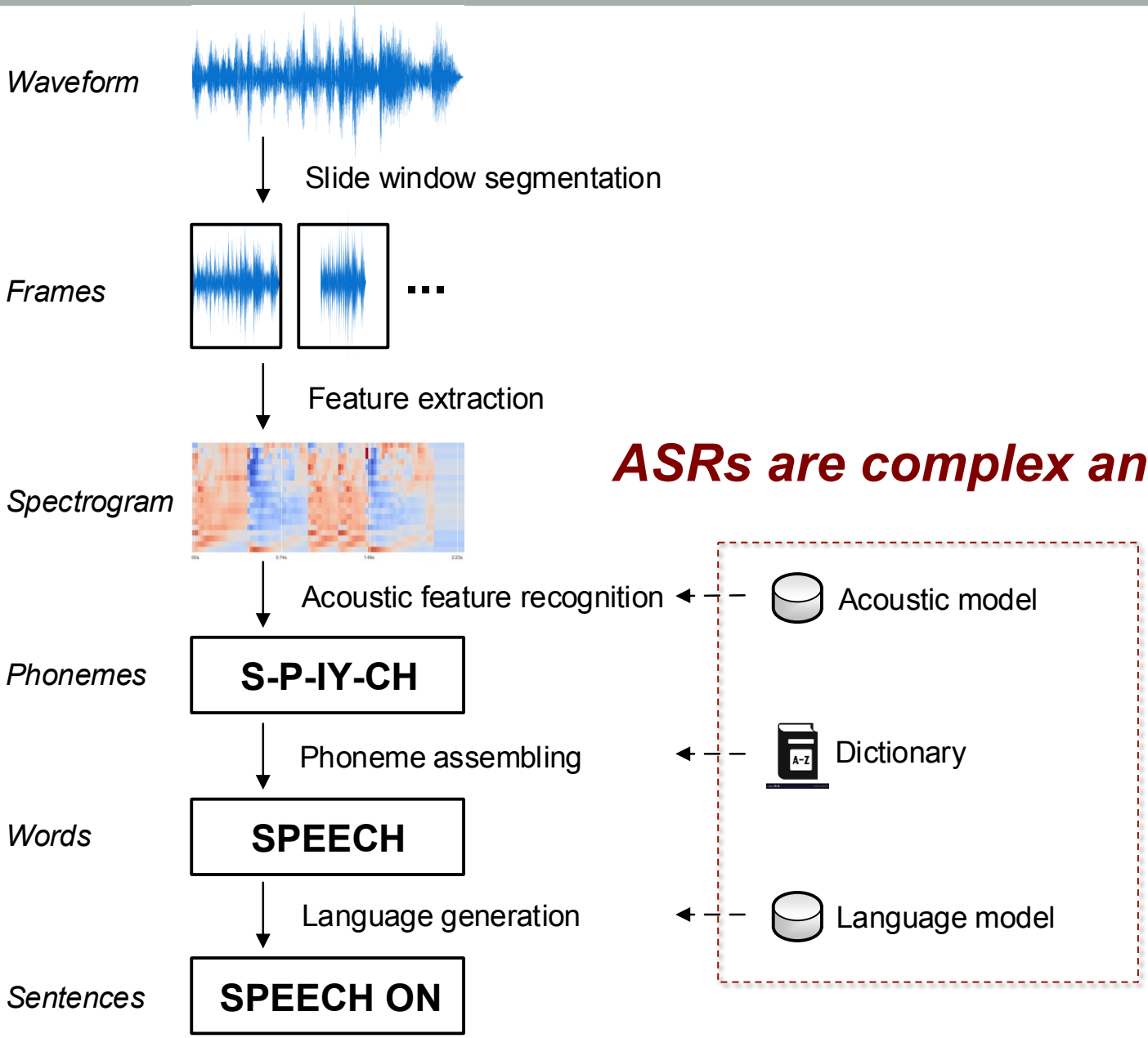
“Open the front door”

- **What is unique about Audio Adversarial Examples (AEs)?**
- **How to detect existing Audio AEs?**
- **How to detect future Audio AEs?**

ASRs Are Ubiquitous

- Automatic Speech Recognition: convert speech to text
- Voice provides a convenient interface for HCI
  - Microsoft, Apple, Google, Amazon
  - Smart phones, homes, cars, etc.
- Playing a popular YouTube song may open your front door





ASRs are complex and diverse

Transferability of Audio AEs

- Audio AE generation methods
  - White-box: internals of the ASR are needed [Carlini & Wagner, 2018]
  - Black-box: only the outputs of the ASR are needed [Alzantot et al., 2018; Taori et al., 2018]

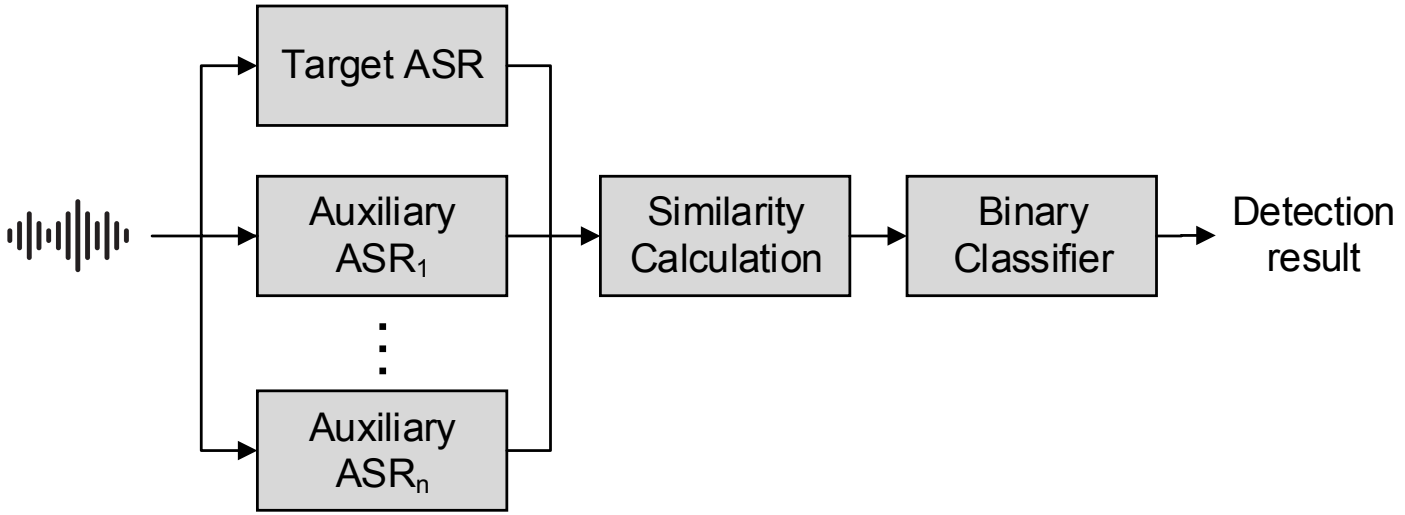
- Transferability of audio AEs is still an *open question* [Carlini & Wagner, 2018]
  - NNs in ASRs have a large degree of non-linearity
  - ASRs are diverse

- **What is unique about Audio Adversarial Examples (AEs)?**
  - ASRs are complex and diverse
  - Transferability of audio AEs is currently poor
- **How to detect existing Audio AEs?**
- **How to detect future Audio AEs?**

Our Idea

- ***Background***: Multiversion Programming (MVP)
  - Multiple programs are independently developed following the same *specification*
  - Such that bugs are usually *not* shared => an exploit that compromises one program is ineffective for other programs
  - Run these programs in parallel, and use voting
- ***Main idea***: MVP-inspired audio AE detection
  - All ASRs follow the same specification: convert speech to text
  - Run multiple ASR systems in parallel
  - If the ASRs generate similar results => the input is benign
  - If the ASRs generate dissimilar results => the input is an AE

System Design



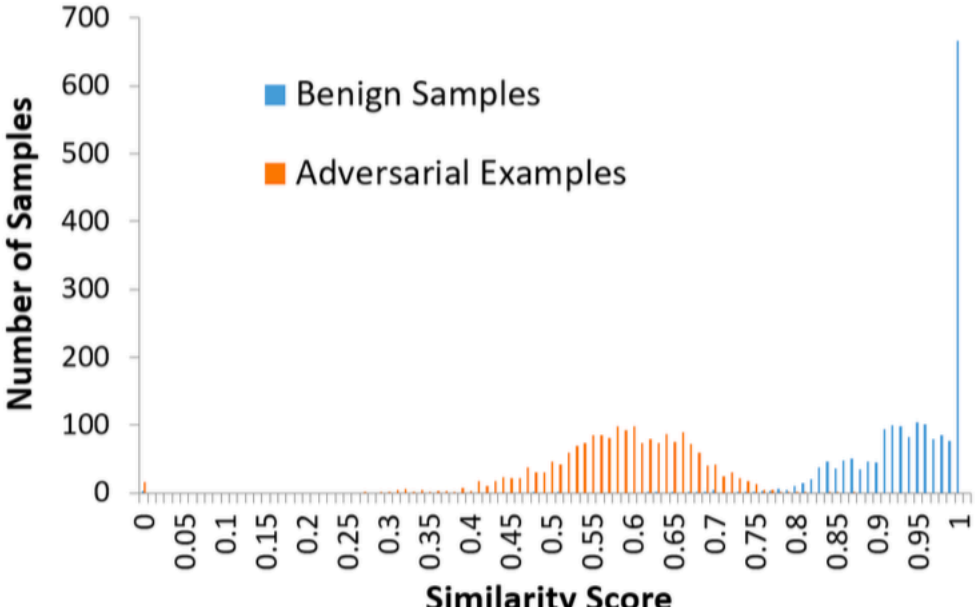
- Target ASR: the ASR targeted by attackers; denoted as *T*
- Similarity calculation
  - Given *n* auxiliary ASRs, *n* similarity scores are calculated
  - *Similarity score: sim(T(input), ASR_i(input))*
  - Phonetic encoding is used, such that *sim ("pear", "pair") = 1*
- Binary classifier: a simple SVM

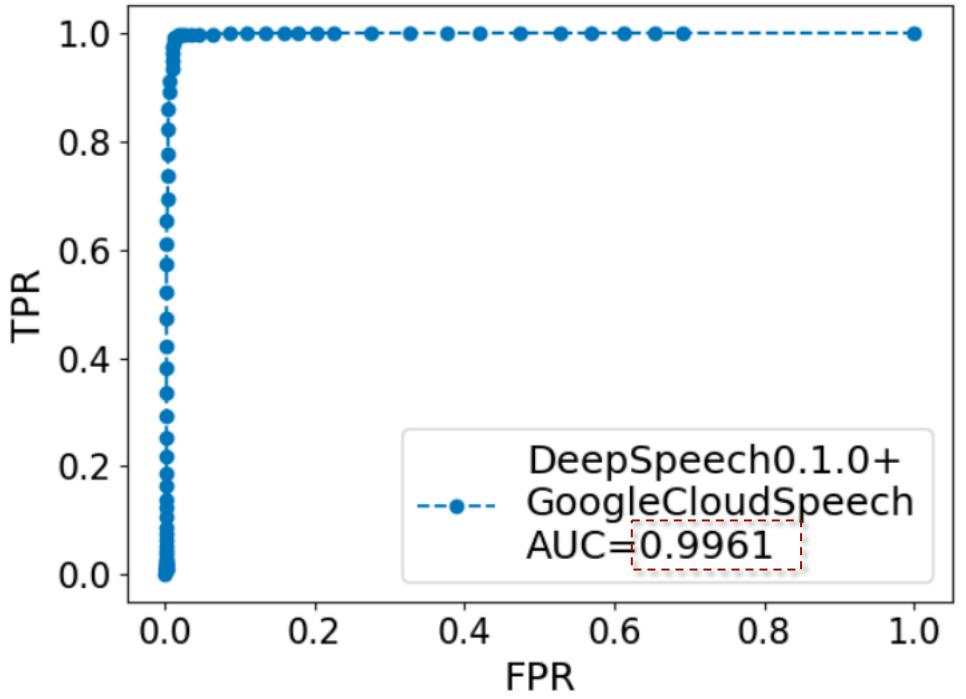
Evaluation Settings

- Target ASR
  - DeepSpeech v0.1.0 (*DS0*)
- Auxiliary ASRs
  - Google Cloud Speech (*GCS*)
  - Amazon Transcribe (*AT*)
  - DeepSpeech v0.1.1 (*DS1*)
- Various combinations exist
  - E.g., if *GCS* and *AT* are used as the auxiliary ASRs, it is denoted as

    DS0 + {GCS, AT}
- Dataset
  - 2400 benign audio samples randomly selected from *LibriSpeech*
  - 2400 AEs = 1800 white-box AEs + 600 black-box Aes

For example, Google Cloud Speech used as the single auxiliary ASR, i.e., DS0 + {GCS}





Detection Accuracy (5-fold cross validation)

When a single auxiliary ASR is used,
the accuracy is 99.56 (using DS1), 98.92% (GCS), 99.71% (AT)

Dose **false positives** increase when there are more auxiliary ASRs?

No, as more "evidences" are present by extra ASRs

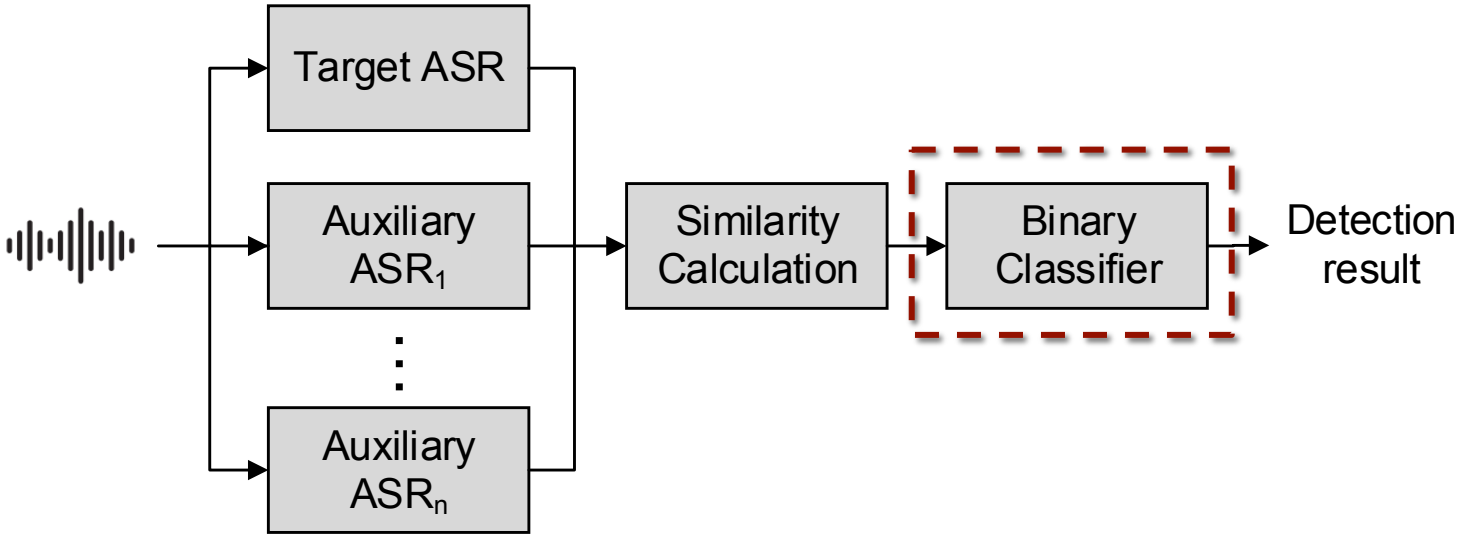
Classifier	Metrics	System			
		DS0+{DS1, GCS}	DS0+{DS1, AT}	DS0+{GCS, AT}	DS0+{DS1, GCS, AT}
SVM	Accuracy	**99.75%**	**99.86%**	**99.82%**	**99.88%**
	FPR	**0.29%**	**0.08%**	**0.08%**	**0.04%**
	FNR	0.21%	0.21%	0.29%	0.21%

- **What is unique about Audio Adversarial Examples (AEs)?**
  - ASRs are complex and diverse
  - Transferability of audio AEs is currently poor
- **How to detect existing Audio AEs?**
  - A Multiversion Programming (MVP) inspired approach
  - Accuracy 99.88%
- **How to detect future Audio AEs?**

In future, attackers may be able to generate transferable audio AEs.

Will this totally defeat this detection approach?

Or, can our approach do better, say, *proactively fight transferable AEs?*



- **Insight 1**: the binary classifier actually is *not* trained using AEs, but using their corresponding similarity scores
- **Insight 2**: the concept of *hypothetical transferable AEs*
  - A hypothetical AE = $\{s_1, s_2, \ldots, s_n\}$
  - If an AE can fool both the target ASR and an auxiliary ASR_i, we assign a **high** similarity score for s_i; otherwise, a low one

- **How high is "high"?**
  - A transferable AE that can fool multiple ASRs will make the ASRs agree on the injected malicious command, just like they agree on a benign sample
  - So we use the scores of 2400 benign samples to construct a pool of high scores

Type	MAE AE	# of MAE AEs
Type-1	$AE(\texttt{DS0}, \texttt{DS1})$	2,400
Type-2	$AE(\texttt{DS0}, \texttt{GCS})$	2,400
Type-3	$AE(\texttt{DS0}, \texttt{AT})$	2,400
Type-4	$AE(\texttt{DS0}, \texttt{DS1}, \texttt{GCS})$	2,400
Type-5	$AE(\texttt{DS0}, \texttt{DS1}, \texttt{AT})$	2,400
Type-6	$AE(\texttt{DS0}, \texttt{GCS}, \texttt{AT})$	2,400

- E.g., *AE(DS0, DS1)* means that the hypothetical MAE (multi-ASR-effective) AE can fool both DS0 and DS1
- We aim to build a **comprehensive** system that detects all the 6 types of transferable AEs
  - Train the system using only type-4, type-5, and type-6 AEs
  - 97.22% accuracy for type-4,5,6 AEs
  - 100% accuracy for type-1,2,3 (and all the genuine AEs)

Overhead

- DS0 + {DS1}
- 8.8 seconds for DS0 to recognize a sample on average
- Delay incurred by our system: 0.065s, that is, 0.74%

Contribution and Limitation

- Empirically investigated the transferability of audio AEs
- A simple but highly effective audio AE detection technique inspired by Multiversion Programming
  - **Accuracy 99.88%**
- ***Proactively*** trained a model that defeats transferable audio AEs even before they exist
  - A giant step ahead of attackers
- *Limitation*: the detection technique fails if the host text and the malicious text are very similar
  - However, existing AE generation methods claim that ***any*** host audio may be used to embed a malicious command
  - Our detection ***dramatically*** reduces this attack flexibility

All the datasets, code and models have been open-sourced

https://github.com/quz105/MVP-audio-AE-detector

Contact: Qiang Zeng (qzeng@cse.sc.edu)

Questions?