An Improved Approach to breaking Audio CAPTCHAs using Noise Removal and Deep Learning

Gautam Krishnan
University of Illinois at Chicago
Chicago, IL, USA
gkrish3@uic.edu

Varshini Sampath
University of Illinois at Chicago
Chicago, IL, USA
vsampa3@uic.edu

Iasonas Polakis
University of Illinois at Chicago
Chicago, IL, USA
polakis@uic.edu

ABSTRACT

CAPTCHAs are challenge-response tests to distinguish legitimate human users from bots. CAPTCHAs based on reading and perceiving text or images restrict blind and partially sighted users from freely accessing the web. Thus, audio CAPTCHAs were introduced as an alternative to visual challenges to make the web friendly and equally accessible to visually impaired people. Most commercial and non-commercial CAPTCHA systems now offer an alternative audio challenge to their more popular visual challenges.

We conducted a large-scale study of audio CAPTCHA systems in the wild, which we believe was the first study of its kind, and built light-weight automated solvers for each one of them using off-the-shelf speech recognition services. We found that 6 out of the 8 CAPTCHA systems that we analyzed were vulnerable to this attack with Google's No-CAPTCHA reCAPTCHA being the most vulnerable(98.3%).

To further improve the accuracy of our automated solvers, in this paper, we build an improvised system that employs audio processing techniques specific to each CAPTCHA system using an open-source tool called Audacity. We evaluate our improvised solvers with 1000 challenges each and observe a significant increase in accuracy with a less than 25s latency for noise reduction. We are also building an offline solver for SecureImage, the strongest of the 8 audio CAPTCHA systems we analyzed, with Intel's open-source deep learning library for speech recognition.

1. INTRODUCTION

CAPTCHAs are completely automated tests that tell humans and computer programs apart. Since their inception in 2000, they have been extensively used by websites to prevent spam account creation and messages, but have gained widespread popularity and use in this era of social networking. They have now become a standard security mechanism employed by all major websites.

CAPTCHAs have long relied on the ability of humans to perform tasks based on visual recognition, cognition and perception easily. So simple text-based or image-based challenges served to distinguish humans from bots. However, a research by Goodfellow et al. at Google recently showed that today's Artificial Intelligence technology can solve even the most difficult variant of distorted text at 99.8% accuracy. Thus distorted text, on its own, is no longer a dependable test. Sivakorn et al. [9] demonstrated an extremely effective system that solves the state-of-the-art ReCAPTCHA's image based CAPTCHAs with an accuracy of over 70% and Facebook's image CAPTCHA with an accuracy of over 83%.

Figure 1: A few speech recognition services.

This led us to test the security of audio based CAPTCHA systems by major CAPTCHA providers and websites. Though audio CAPTCHAs were introduced for reasons of accessibility to visually impaired people, the option to listen to an audio challenge instead of solving the standard text or image based challenge exists for normal users too. In other words, they are accessible to bots too. For this reason, we evaluated the security of audio CAPTCHAs from 8 popular CAPTCHA providers and built our own automated solvers using existing speech recognition services. In this paper, we propose to improve the accuracy of our automated solvers using noise reduction techniques.

2. PREVIOUS WORK

Previous work in breaking audio CAPTCHAs involved the use of *Hidden Markov Models (HMMs)* or *Artificial Neural Networks* (ANNs) to perform automatic speech recognition on segments of the CAPTCHA. To apply these classification methods, initially the input signal is first segmented into a series of overlapping frames to extract feature vectors. The three feature representations that are currently relied upon are Mel-Frequency Cepstral Coefficiants (MFCC), Perceptual Linear Prediction (PLP), and relative spectral transform-PLP (RAS TA-PLP). Using HMM-based ASR[6], Hendrik et al achieved an accuracy of 62.8% with Google's reCAPTCHA (2014 version). A later paper by Hendrik et al also exploited semi-supervised learning techniques as a cost-effective alternative to break audio CAPTCHAs[7].

Tam et al[1, 2] created their own classification system, training their data set on scraped CAPTCHAs.[1] Bursztien et al[8] adopted a two-stage approach to solving audio CAPTCHAs, first segmenting the audio file into parts and then getting the individual transcriptions for each of the parts using a classification algorithm. They evaluated their system which they called DeCaptcha, against six audio CAPTCHA systems and broke five of them, all with more than 40% accuracy.

Other previous work, relied on modifying existing off the shelf tools such as the CMU sphinx speech recognizer[5], using machine learning to improve its accuracy to 75%. It was observed that CMU Sphinx without the help of machine learning was able to give an efficiency of 9.6% with the TIDIGITS model, and 28.9% with the HUB4 model.

Thus, all these systems found performance of off-the-shelf tools unreliable and inefficient and relied on machine learning techniques to build their solvers. Our approach exploits features in existing speech recognition services to solve audio CAPTCHAs with a higher level of accuracy than all previous studies. We are looking at noise reduction and deep learning as avenues of improvements over our obtained results.

3. BREAKING AUDIO CAPTCHAS - OUR APPROACH

We did a large-scale study of the security of audio challenges from eight different CAPTCHA providers and websites including Apple, BotDetect, CaptchasNet, Live, Google reCAPTCHA v1, Google reCAPTCHA v2, SecureImage and Telerik. We built web crawlers to scrape the audio challenges from these services, solve them using solvers based on speech recognition services and input back the results in the CAPTCHA textbox. We evaluated our system initially with 50 CAPTCHAs each to identify the best solver and then tested the best solver for each CAPTCHA system with 1000 challenges.

The results of our experiments are given in Fig 2. We found that all of the 8 CAPTCHA systems that we analyzed could be broken with our solvers, despite background noises and the rate limiting that some providers implement. This includes the most popular CAPTCHA system in use - Google's reCAPTCHA (98.3%) and BotDetect (24.6%), which is popularly used in government websites.

In this paper, we propose to improve the accuracy of our automated solvers by using audio processing techniques. We identify a sequence of actions like Amplification, Noise Reduction, Low-pass audio filtering, etc that work best for each CAPTCHA system in reducing background noise and making the audio more discernible. For this, we use a free open-source cross-platform audio editor called Audacity. We build Action Chains that describe the set of actions specific to each CAPTCHA system and pass our audio file obtained while solving the CAPTCHAs, real-time, to Audacity. Audacity then cleans up the audio using predefined Action Chains and gives the denoised file back to our system. We observe that the accuracy of our solvers improves significantly for certain CAPTCHA systems like Telerik, Apple and Captchas.net.

CAPTCHA Service	IBM - US	IBM - UK	Wit	Google - US	Google - UK
Apple	16% (50)	14% (50)	0% (50)	58% (50)	44% (50)
BotDetect	4% (50)	12% (50)	4% (50)	10% (50)	26% (50)
CaptchasNet	2.16%	25.67%	2.69%	3.50%	36.81%
Live	0%	0%	0%	0%	0%
Recaptcha V1	62% (50)	16.18%	25%	8%	14%
RecaptchaV2	95.80%	67.20%	67.10%	98.30%	81.60%
SecureImage	0%	0%	0%	0.10%	3%
Telerik	64.70%	11.74%	32% (50)	78% (50)	60% (50)

Figure 2: Results from our initial approach, before noise reduction.

Finally, we build an offline neural network based classifier with Intel's Deep Learning system for speech called the **deepspeech**. We initially build this for SecureImage, identified to have the strongest audio challenges among the eight, based on the very low rate of accuracy from our experiments (3%). We plan to extend this idea and create classification models for each of the other CAPTCHA systems as a possible future work.

Since SecureImage is an open-source CAPTCHA system, we create a training dataset of 1000 audio sample files mixed with their background noises and train our deepspeech classifier with the truth values. We train our system to create a deep learning model and evaluate the model with a test dataset.

We consider the following to be our main technical contributions in this paper:

- We present a novel, low-cost approach to breaking audio CAPTCHAs and improving their accuracy by audio processing techniques.
- We evaluate our improvised system against all eight services by breaking 1000 CAPTCHAs each using the best solver identified after denoising and observe improvement in accuracy.
- Our biggest contribution was in proving that audio CAPTCHAs can be solved by open-source services and audio tools that are readily available in the market, with very less effort. This is because speech recognition services developed by big companies like Google, IBM Watson, etc, themselves do a lot of audio processing, noise filtering and machine learning in the background.
- We are also in the process of creating a neural-network based classifier for speech recognition that can work as an offline audio CAPTCHA solver.

4. LITERATURE REVIEW

This section gives a brief overview of what constitutes noise when it comes to human speech, a classification of the

different kinds of noise and the audio processing techniques we used in our system to clean up the audio files.

4.1 Sound

Sound is created by alternate compression and decompression of particles of the air. This causes the air pressure to fall and rise in the form of waves. Frequency (pitch) and amplitude (loudness) are the two main characteristics of sound. **Frequency** is the number of times that the air is compressed and decompressed in a second, and is measured in cycles per second, or Hertz (Hz). Low frequency produces a low pitched, bass sound. High frequency produces a high pitched, whistle-like sound. Human ears respond to frequencies between 20Hz and 20,000Hz. **Amplitude** is the amount of sound energy reaching the eardrum, and is measured in decibels (dB). The safe range of amplitude for a human ear is 0 to 140dB.

4.2 Noise in Speech

The human voice produces frequencies between 500Hz and 2,000Hz. Background noises have higher frequency levels than speech. The comprehension of speech is affected by both amplitude of the background noise and the amplitude of the voice itself. The average amplitude of a human voice in a room at a distance of one meter lies within the following ranges:

- Conversation 60-65dB
- Dictation 65-70dB
- Calling out 80-85dB

The general background noise level must be at least 10dB lesser than these levels if the sound of the voice is to be heard clearly.

4.3 Classification of noise

Noise can be broadly classified into 2 kinds - Internal noise and External noise.

Internal noise is noise generated within the receiver or communication system. Colour noises like white noise, pink noise, brown noise,etc are examples of internal noises with specific characteristics, that can sometimes be manually added to mask other disrupting background noises. **External noise** is noise generated external to the system. So, all kinds of background noise come under this category.

4.4 Audacity

We looked for a free, open-source, light-weight digital audio editing tool that was available in the market because our goal was to build a low-cost Audio CAPTCHA breaker. For this reason, we selected Audacity. Audacity is a free digital audio editor and recording tool that is available for Unix, Linux, OSX and Windows platforms. It has a number of built-in audio processing mechanisms that can be fine-tuned to our needs.

4.5 Audio processing techniques

There are several audio processing techniques that are commonly used to remove noise and make recordings clearer in audio post-processing. All these techniques are available a single-click away in Audacity. So we exploit these functionalities without having to worry about the algorithm and the implementation. The following are some of the audio processing techniques that we use in this paper.

- **Amplification** : Amplification is the process of increasing the amplitude of the audio by a certain amount or ratio. It just makes the audio sounding louder.
- **Equalization** : Equalization is the process of adjusting the frequency response in an audio after recording. It is generally used to make certain sounds in an audio more prominent than others. We specifically use it to make vocal sounds louder and the background noises quieter.
- **Normalization**: Normalization is done to set the peak amplitude of a track. A normalizer identifies the peak amplitude in the track and the maximum allowed amplitude and calculates gain as the difference between these two values. It then applies this gain for the entire track.
- **Noise Gate** : Noise gates are used to remove constant background noises like hiss sounds, murmurs, leaf rustles, etc. It does this by allowing signals above a threshold amplitude value to pass through the gate, while blocking everything below.
- **Noise Filter** : A noise filter is used to remove frequencies that do not fall within a certain range. To accomplish this, it uses 2 filters - a low pass filter that attenuates frequencies lower than particular value and a high pass filter that attenuates frequencies higher than a particular value.
- **Noise Reduction** : This technique samples a piece of the audio as a noise profile and applies a reduction in amplitude to the parts of the audio that has this noise.

5. SYSTEM OVERVIEW

Our system consists of a Python web crawler that loads the required page, finds the audio element in the page and downloads the challenge. It then converts the audio file into the required audio format and hands it over to Audacity that applies the appropriate Action Chain for the CAPTCHA system. The action chain or audio processing chain used for each CAPTCHA service will be explained in further detail in Section 5. The connection between Audacity and our crawler is established through open-source libraries like *ldtp* or *pywinauto*.

Once Audacity completes the audio processing, control is transferred back to our system that connects to the speech recognition web services. When a list of alternative transcriptions is returned, our script selects the most appropriate one among them. It then enters the response into the response box using Selenium, and the challenge is submitted.

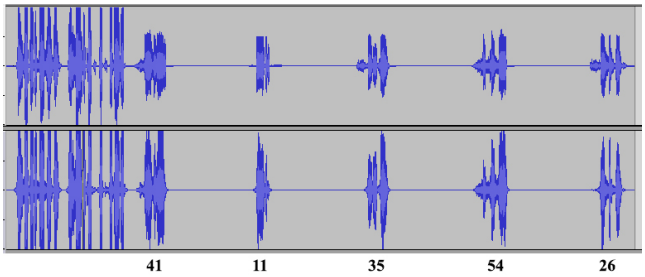

Figure 3: Waveforms of Apple's CAPTCHA, before and after processing, representing "41 11 35 54 26".

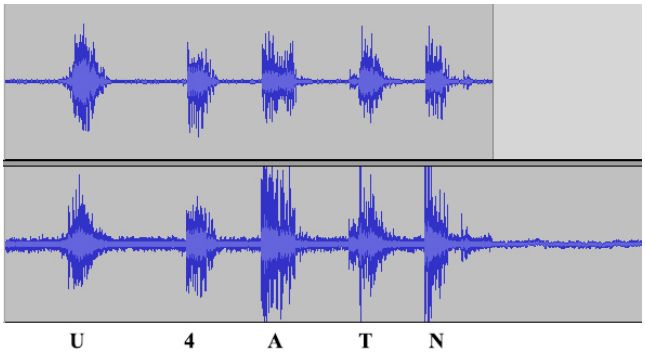

Figure 4: Waveforms of BotDetect's CAPTCHA, before and after processing, representing "U4ATN".

Once it is submitted, an element on the page is generated which tells the user if the submitted response was correct or not. We look for this element on the page and gather this result. We next store these results in a CSV file along with the file name and the time it took to solve each challenge.

6. ANALYSIS OF AUDIO CAPTCHA SYSTEMS

This section describes the 8 audio CAPTCHA systems that we analyzed in our previous work, the properties and kinds of noise that make it less discernible to speech recognition services and our approach to overcome those challenges to achieve better accuracy. The results of these experiments are given in Section 8.

For each CAPTCHA system, the sequence of audio processing techniques, referred to as Action Chains by Audacity, was obtained on a trial-and-error basis with a huge number of CAPTCHAs before the experimental evaluation. We selected cut-off frequency values, amplification factors and filter ranges using spectrum plots of audio samples from each CAPTCHA system.

6.1 Apple

Apple uses its own CAPTCHA system for authenticating humans during apple ID creation. The audio challenge has a male voice spelling out two-digit numbers (from 11 to 99) at equal time intervals. The length of the challenge randomly varies between three to five such numbers. With each number that is spelled out, there is a background

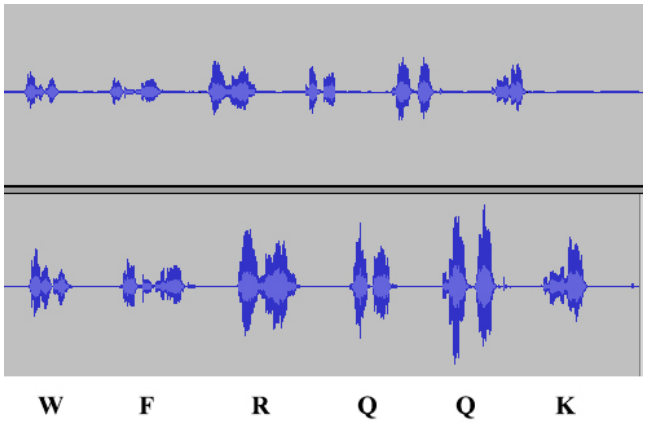

Figure 5: Waveforms of captcha.net's CAPTCHA, before and after processing, representing "WFRQQK".

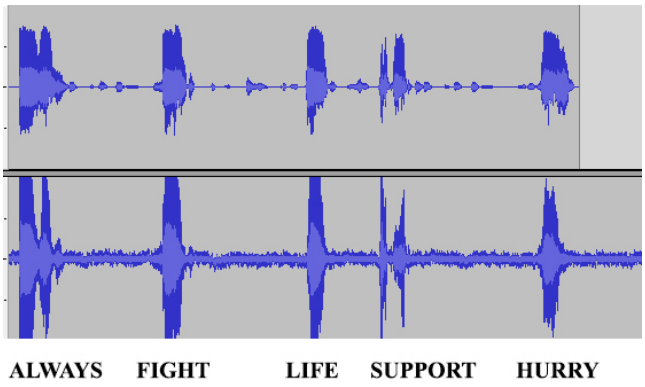

Figure 6: Waveforms of Microsoft's CAPTCHA, before and after processing, representing "always fight life support hurry".

noise that makes distinguishing between numbers like fifty-two and sixty-two difficult even for humans. The nature of the noise is external and is composed of sounds of children laughing and talking. There is also a very small internal noise throughout the audio, but we observed that it did not do much to disrupt the speech recognition. The system does not give any margin of errors to users.

The waveforms of Apple's audio CAPTCHA, (representing the challenge "41 11 35 54 26") before and after denoising are shown in Figure 3. To denoise the audio, we equalized the audio using Audacity's standard equalizer with a -18dB bass cut and +6dB treble boost. We then passed it through a noise gate and low-pass filter to block frequencies higher than 3000Hz and -18dB and frequencies lesser than 500Hz, based on analysis from the spectrum plots. In the resulting audio, we applied noise reduction with a noise profile created for Apple's audio captcha in particular and finally the cleaned up audio was amplified and mixed with a 0.5dB pink noise signal.

6.2 BotDetect

BotDetect is a CAPTCHA system commonly found in several US government websites like the U.S Department of

State, Supreme Court of United States, etc. as well as sites like Morgan Stanley, Dell and Intuit. The audio challenge consists of a single male voice spelling out 6 random character that includes alphabets and numbers. It is an open-source CAPTCHA system that allows developers wanting to incorporate BotDetect's CAPTCHA to their websites to play around with the length of the challenge and the kind of noise used in the background. The standard length of challenges is 4-6 characters and the kinds of noises randomly generated in the background include industrial noises, radio, pulse, hive, robot, workshop and others. The system does not have any margin for errors.

To denoise such a CAPTCHA system with dynamically generated background noises was a real challenge. We created a general rule that best worked for many of these noise types, if not all. We filtered the audio file using a high-pass filter to cut off frequencies below 1000Hz, equalized it using the standard values that work best for human speech, amplified by +4dB. Finally, we found that reducing the pitch of the speaker by one scale down (-5.6%) helped the speech recognizers in better identifying the words.

6.3 captchas.net

Captchas.net is also an open-source CAPTCHA system that is integrated in CMS tools like Joomla and Plone and popular blogs to prevent spam activity. The audio challenge consists of 6 unequally-spaced words spelled with the NATO phonetic alphabets. Users are expected to enter the first alphabet of each of the NATO phonetic words that they hear. For example, the challenge "WFRQQK" in Figure 5 gets spelled out as "Whiskey Foxtrot Romeo Quebec Quebec Kilo" and the user is expected to recognize these words and input "WFRQQK" in the text box given. There is no external background noise that is added. But the voice seems to be robotic and words are of different intensities and pitches which makes it difficult for speech recognition systems to work. This system too requires users to get all 6 characters right.

In an attempt to help speech recognition services to get past the variable space intervals issue, we reduced the tempo of the audio by 20% and then amplified it by a ratio of 2.26. We then normalized the audio file by a factor of -2.0 to amplify bigger signals and soften the smaller ones. We then truncated the silence caused by reducing the tempo and mixed it with a small pink noise of 0.8dB to get a cleaner version of the audio.

6.4 Microsoft Live

Microsoft uses a self-designed CAPTCHA system in its account creation page. The audio challenge consists of random English words v, varying in length (5-7 words), pronounced by a man and a woman, some of which are not perceivable even to humans. Since the words are not from a specific corpora of words, users find it difficult to identify words and often have to make hard choices between words like "quiet" and "quite", and "sure" and "shore". The audio also has a very unique kind of external noise with a female noise speaking in a language different from English in the background. The system expects users to get all the words right.An example of Microsoft's CAPTCHA is shown in Figure 6.

To denoise this kind of CAPTCHA, we equalized the audio with -18dB bass cut and +8dB treble boost and then passed it through a noise gate to ward off signals higher than Hz frequency. We sampled a piece of the background noise and using this as a noise profile, ran a noise reduction algorithm and equalized it again. To mask the noise that was left over, we added a strong pink noise of 0.15dB, higher than the value we used for other systems. This gave us a much cleaner sounding audio with the background noise masked by a clean pink noise. The comparison between the clean and the denoised audio signals can be seen in the image.

6.5 Google's reCAPTCHA v1 (Dec 2014 version)

Google introduced reCAPTCHA with the title "Stop Spam. Read books." in the year 2011 and over the years kept improving their CAPTCHA system for reasons of usability as well as security. The kind of audio CAPTCHAs that Google used also kept changing. We tested the audio challenge that was present in reCAPTCHA v1(Dec 2014). This version of audio reCAPTCHA had 9-10 digits spoken out by different voices in different accents, all unequally spaced. Some digits had some background noise associated with them and the noise was hissy and internal in nature. It also allowed upto one wrong digit by the user.

We couldn't do much to denoise the reCAPTCHA audio files as techniques like Noise reduction and noise gating further decreased the accuracy of the speech recognition services. So we did a simple slowdown of the audio by reducing the tempo by -25% and then amplifying the intensity of the sound by a ratio of almost 3.0.

6.6 Google's reCAPTCHA v2 (Jan 2017 version)

The version 2 of Google's reCAPTCHA was much simpler than its predecessor. It had just 5 digits, spelled out in equal intervals with little to no noise at all. We had a very high accuracy of 98.3% using Google's Speech API, even before denoising. The system also allowed upto 2 incorrect digits by the user. Since there was little to no noise in the audio files, we did an amplification of the audio followed by reduction in tempo and then equalization.

6.7 Securimage

Securimage is a free open-source CAPTCHA service that is readily available for integration in PHP. It is being used in many government websites and portals, since it is free and open-source. The CAPTCHA challenge consists of 6 characters with alphabets and digits. The audio has a lot of noise and inconsistent intervals between the characters spelled out. The kinds of noise vary from airport noises, crowds talking, children playing and laughing, industrial noises, musical instruments etc. The amplitude of the noise is so high that sometimes the CAPTCHA's audio gets lost within the noise. This system too does not have any margin for error and expects users to get all 6 characters right.

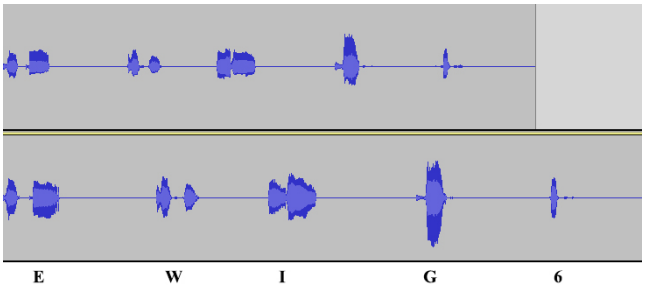

Figure 7: Waveforms of Telerik's CAPTCHA, before and after processing, representing "EWIG6".

To clean these audio files, we created a noise profile with all the open-source noise files that Securimage offers and then did a -18dB noise reduction, a reduction in tempo by -20% and then amplified the signals to make them louder. But the combination of noise files used for each CAPTCHA varied so widely that a general noise profile did not work very well for all captcha challenges.

6.8 Telerik

Telerik is a company that offers software solutions for web, mobile and desktop application development. Their CMS Sitefinity has many customers including several colleges in University of Illinois. Telerik offers its own CAPTCHA system with an audio alternative. The challenge has a female voice speaking out 5 words that may be NATO words or numbers. The words are spaced at fairly equal intervals, but the tempo at which each word is spoken varies. The nature of the noise is internal with words getting slightly cut and distorted at times.

The noise reduction profile for Telerik consists of an equalization step with a -18dB bass cut and a +8dB treble boost, followed by an amplification and noise gating to gate only frequencies above 1400Hz. The resulting signal is then passed through a low-pass filter to remove frequencies lesser than 300Hz and then slowed down the audio by -20%. Finally the signal is normalized by -2dB and this resulted in very clean version of the audio that performed better with speech recognition services.

7. EXPERIMENTAL EVALUATION

This section describes the experiments we ran to evaluate our system against each CAPTCHA service. Initially, we ran a live attack with our improvised system with 50 challenges for each CAPTCHA system. We did this attack individually for all 5 solvers - IBM Watson US Accent, IBM Watson UK Accent, Wit.ai, Google Speech API US Accent and Google Speech API UK Accent. The details of the live attack are described in brief detail below.

Based on the results of these experiments, we identified one solver to be the best for each CAPTCHA system. Then we did a larger attack with 1000 CAPTCHAs for each service with the identified best solver and recorded the results of whether our response got accepted or not. For each challenge we solved, we also recorded the time it took to solve that challenge.

- **Apple** : To test Apple's CAPTCHAs, we launched a live attack on the Apple ID creation page. We ran our crawler on the page multiple times, filling all the text fields and dropdown boxes with random values and then downloading the audio challenge from chrome's media-internals page. The downloaded file was then sent to Audacity to get the cleaner version using the Action chain described in the previous section and the resulting .wav file was sent to the speech recognition services for transcription. After the results were obtained and entered back into the response box, if the CAPTCHA was solved properly, a 2-factor authentication dialog box was displayed. The presence of this dialog box with class *step-verify-code* was checked for and the results were logged accordingly.

- **BotDetect** : To test our system against BotDetect, we attacked the demo page of BotDetect present in *www.captcha.com*. The demo page randomizes the length of the audio challenge(4-6) and the type of noise(10 noise files) added in the background so that we get a completely different kind of audio challenge each time the page is loaded. We downloaded the challenge from this demo page and solved it using our solvers after denoising. We looked for the presence of "Correct!" or "Incorrect!" in the div next to the Submit button after it is validated.

- **captcha.net**: captchas.net is an open-source CAPTCHA system. So we hosted this service on our test page and performed our live attack on this URL. We scraped the audio MP3 file from the loaded URL and cleaned it using Audacity and then solved it using our solvers. The resulting page either displayed "*You entered the wrong password. Aren't you a human?*" or "*Your message was verified to be entered by a human*". We looked for this element and then validated our solver.

- **Microsoft Live** : The Microsoft account creation page is where we launched our attack. However, we did not perform a live attack for this one because there was no way to validate the CAPTCHA before the page was submitted. If we did submit the page, there were no further steps. A spam account would get created if the response to the CAPTCHA was right. So we decided to do an offline attack by just scraping 100 CAPTCHA challenges from Live's page and then manually listening to the audio files to get the transcription. To get a valid truth value, 3 people were asked to listen to the 100 audio files and write down their transcriptions manually. When our solvers gave us transcriptions after denoising,

- **reCAPTCHA V1** : The 2014 version of Google's reCAPTCHA is still available in a demo page offered by Google in the URL *https://www.google.com/recapt-cha/demo/*. We performed a live attack on the demo page. Once our system cleaned up the audio file and got the results back from the solvers, we looked for the presence of the text "Correct" and "Incorrect" on the top of the page.

- **reCAPTCHA V2** : We tested our system against Google's NoCAPTCHA reCAPTCHA (Jan 2017 version) on an incognito mode. We clicked on the check-

box CAPTCHA and clicked on the Audio icon on the Image challenge iframe to get to the Audio CAPTCHA. We then downloaded the .mp3 file and solved it using our system. We looked for the presence of tick mark to verify that the challenge was solved correctly. To get past the rate limiting that Google had, we found an implementation bug in Google's reCAPTCHA that considered requests with unique user-agent strings as trust-worthy. So we generated unique user-agent strings with current timestring value.

- **Securimage** : Securimage is again an open-source CAPTCHA system. So we hosted the service on our test page and performed our live attack on this URL. The URL when loaded dynamically constructed an audio CAPTCHA from the 26 files for 26 alphabets and the 6 noise files it had. We scraped this audio WAV file from the loaded URL and cleaned it using Audacity and then solved it using our solvers. The resulting page after submission either displayed "*Incorrect security code entered*" or "*The CAPTCHA was correct.*". We looked for this element and then validated our solver.

- **Telerik** : Telerik offers a demo page for its Audio CAPTCHAs for ASP.NET developers. We performed a live attack on this page, scraping the audio files and cleaning it using Audacity using an Action Chain. The resulting WAV file was sent to the speech recognition APIs and the transcription was obtained. The presence of the text "Page submitted successfully" or "Page not valid" was checked for and recorded.

8. RESULTS

We demonstrated that our improvised system is highly effective against Apple's CAPTCHA system resulting in a 22% increase in accuracy for Google UK solver. This yielded us our best solver for Apple with 66% accuracy. We got a final accuracy of 60% when testing with 1000 CAPTCHAs. It also performed well with Telerik's CAPTCHA system, improving the accuracy of IBMWatson US Accent solver from 64% to 88%. When tested with 1000 CAPTCHAs, we got an accuracy of 90%. All the results of our experiments are given in Figure 8 .

We observed that denoising helped all CAPTCHA systems except reCAPTCHA and Securimage. The quality of the audio file decreased after denoising for reCAPTCHA because the background noise that was added was of very short-length and difficult to remove. In case of Securimage, there was too much background noise and the amplitude of the background noise was as high was the actual signal. Thus removing it was very difficult. Also, on the whole, Wit.ai solver performed better before denoising than after it.

Microsoft Live was hard to crack because they do not have a corpora of words that they construct audio challenges from, unlike others that give out digits, alphabets, NATO words or a combination of any of these. We also showed that our attack was least effective against Securimage, which uses background noise to mask the audio.

CAPTCHA Service		IBM - US	IBM - UK	Wit	Google - US	Google - UK
Apple	Before denoising	16% (50)	14% (50)	0% (50)	58% (50)	44% (50)
	After denoising	12% (50)	6% (50)	8% (50)	64% (50)	66%(50)
BotDetect	Before denoising	4% (50)	12% (50)	4% (50)	10% (50)	26% (50)
	After denoising	16% (50)	16% (50)	4% (50)	8% (50)	4% (50)
CaptchasNet	Before denoising	2.16%	25.67%	2.69%	3.50%	36.81%
	After denoising	26% (50)	30% (50)	0% (50)	6% (50)	38% (50)
Live	Before denoising	0%	0%	0%	0%	0%
	After denoising	10%	3%	1%	1%	2%
Recaptcha V1	Before denoising	62% (50)	16.18%	25%	8%	14%
	After denoising	4% (50)	0%(50)	6% (50)	0% (50)	4% (50)
RecaptchaV2	Before denoising	95.80%	67.20%	67.10%	98.30%	81.60%
	After denoising	86% (50)	60% (50)	12% (50)	82% (50)	66% (50)
SecureImage	Before denoising	0%	0%	0%	0.10%	3%
	After denoising	0% (50)	0% (50)	0%(50)	0% (50)	4% (50)
Telerik	Before denoising	64.70%	11.74%	32% (50)	78% (50)	60% (50)
	After denoising	88%	32%	10%	60% (50)	86% (50)

Figure 8: Results - Accuracy of our solvers with all 8 CAPTCHA systems, after denoising.

9. DEEP LEARNING

Deep Learning is relatively new area in Machine learning that uses many layers of processing units to extract feature information and create a classification model. It is intended to make a system more artificially intelligent by training on huge datasets. We looked at this class of machine learning for creating an offline solver for Audio CAPTCHAs. Since we had very low rates of accuracy for Securimage and Live, we looked to create separate classification models for them.

To do this, we used Intel NervanaSystem's open-source deep learning library for speech recognition called **deepspeech**. The source code for this library can be accessed from their GitHub URL (https://github.com/NervanaSystems deepspeech). Their implementation is based on Baidu SVAIL's Deep Speech 2 in neon [].

We installed deepspeech on a machine with 16GB RAM and an NVIDIA Titan X GPU card. We are in the process of training the system with a dataset of 10,000 CAPTCHA files from Securimage along with their truth values. Once trained, the model can be used to predict transcriptions for audio CAPTCHAs obtained at real-time.

10. FUTURE WORK

As mentioned earlier, we are looking to create our own classifiers with an off-the-shelf open-source deep learning library for speech recognition. We intend to do this for each CAPTCHA system by training with datasets collected from all our previous experimental runs and evaluations. Our focus is mainly towards Securimage and Microsoft Live since we have low accuracy rates for both of them.

We also intend to look into 2 other CAPTCHA systems - Mollom and SolveMedia both of which are being used by some popular websites.

11. CONCLUSIONS

In this work, we demonstrated a novel, low-cost approach to break audio CAPTCHAs. We did this using the off-the-shelf speech recognition APIs and libraries. We also demonstrated how using simple audio processing techniques using an open-source tool like Audacity helps in improving accuracy.

We evaluated the eight audio CAPTCHA systems against the five different solvers that we used and presented the results along with accuracy and the time taken to solve the challenges. We saw how our system faired poorly against Securimage and we also discussed how we could improve our system in the future to increase our attack against that service.

12. REFERENCES

1. Tam, Jennifer, et al. "Breaking Audio CAPTCHAs." NIPS. 2008.
2. Tam, Jennifer, et al. "Improving audio captchas." Symposium On Usable Privacy and Security (SOUPS). 2008.
3. Aiswarya, K., and K. S. Kuppusamy. "A Study of Audio Captcha and their Limitations."
4. Breaking Google's audio CAPTCHA: http://www.networkworld.com/article/2278947/lan-wan/breaking-google-s-audio-captcha.html
5. Bursztein, Elie, and Steven Bethard. "Decaptcha: breaking 75% of eBay audio CAPTCHAs." Proceedings of the 3rd USENIX conference on Offensive technologies. USENIX Association, 2009.
6. Meutzner, Hendrik, et al. "Using automatic speech recognition for attacking acoustic CAPTCHAs: The trade-off between usability and security." Proceedings of the 30th Annual Computer Security Applications Conference. ACM, 2014.
7. DarnstÃd'dt, Malte, Hendrik Meutzner, and Dorothea Kolossa. "Reducing the Cost of Breaking Audio CAPTCHAs by Active and Semi-supervised Learning." Machine Learning and Applications (ICMLA), 2014 13th International Conference on. IEEE, 2014.
8. Bursztein, Elie, et al. "The failure of noise-based non-continuous audio captchas." Security and Privacy (SP), 2011 IEEE Symposium on. IEEE, 2011.
9. Sivakorn, Suphannee, Iasonas Polakis, and Angelos D. Keromytis. "The cracked cookie jar: HTTP cookie hijacking and the exposure of private information." Security and Privacy (SP), 2016 IEEE Symposium on. IEEE, 2016.
10. Amodei, Dario, et al. "Deep speech 2: End-to-end speech recognition in english and mandarin." arXiv preprint arXiv:1512.02595 (2015).