A Survey of Audio CAPTCHAs in the wild and breaking them using Existing Speech Recognition Services

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

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

1. ABSTRACT

CAPTCHA is a backronym for Completely Automated Public Turing test to tell Computers and Humans Apart. As the name suggests, they are automated tests designed to control access to computer systems by distinguishing humans from bots. Since their launch in 2000, websites have used them extensively to prevent bot account creation and spam. There are many types of tests devised for CAPTCHAS in the past. The most common CAPTCHAS are text based, where an image is shown to the users which contains distorted text. Users are asked to read and recognize this text and type the text they see into a textbox that is situated next to the image. Some other common types of CAPTCHAs include tasks of selecting a single or multiple similar images among many heterogeneous images, solving math puzzles, recognizing 3D text, clicking or dragging moving elements, locating text or numbers in a video, etc.

For a long time now, researchers have tested the strength of text CAPTCHAs and using the latest advancements in computer vision, have been able to break text CAPTCHAs with an accuracy of up to 77% [6]. The version 2 ReCAPTCHA by Google employs checkbox and image CAPTCHAs which again, after scrutiny, have been rendered broken with an accuracy of 70.78% [11]. Ever since CAPTCHAs have been introduced, companies have built their own CAPTCHA systems and have used them to prevent bot activity on their web pages. Also, a number of enterprise solutions that offer CAPTCHA systems as a service have since emerged. Although CAPTCHAs have become very common, users often encounter CAPTCHAs which are hard even for humans to solve. As there is very less or no margin for error to prevent bot activity, users often end up making multiple attempts to solve a challenge. It has been shown that CAPTCHA challenges frustrate users and it negatively impacts the experience and traffic of a website [14].

Commercial CAPTCHA services can be utilized by users by signing up with one of the CAPTCHA providers and including a few lines of their code on the target web pages. ReCAPTCHA is one such service offered by Google. Text and image challenges are the most common CAPTCHAs offered by these services which are implemented on web pages to prevent bots from filling web forms automatically. But a drawback with these CAPTCHA systems is that certain groups of people, especially the visually impaired, find it difficult to use them. Thus, audio challenges were introduced specifically to help computer systems be accessible to the visually impaired. The caveat with this approach is that these audio CAPTCHAs open a new ground for exploitation for attackers.

While there has been a lot of prior work done on breaking text and image CAPTCHAs, audio CAPTCHAs have not been studied to such great detail. In this work, we present a survey of all the audio CAPTCHA services that are currently used and show how we used our system to automate the solving of these challenges using existing, off-the-shelf speech recognition services. We found that the latest version of ReCAPTCHA is very vulnerable to our attack.

2. INTRODUCTION

In this work, we consider multiple leading CAPTCHA providers which also provide an optional audio challenge along with the traditional text or image challenge. To solve these challenges, we consider multiple online speech recognition APIs (which we will henceforth refer to as ”solvers”).

A CAPTCHA is a type of challenge-response test used in computing to determine whether or not the user is human. CAPTCHAs based on reading text or other visual-perception tasks prevent blind or visually impaired users from accessing the protected resource. These accessibility limitations of CAPTCHAs made the CAPTCHA providers feel that they must cater to people who cannot perform such tasks. The solution that was devised to tackle this problem was to provide an alternative, special type of challenge in the form of audio which speaks out digits, characters or other types of text (like NATO alphabets or a few common words from English vocabulary) with some background noise. This can be heard by users and solved to pass the Turing test. Despite the fact that Audio CAPTCHAs have been demonstrated insecure, it is essential to provide these services for the visually challenged.

We provide results for our experiments by solving 50 challenges with each CAPTCHA provider-solver pair to determine the best solver. We then solve 1000 challenges for each of the best provider-solver pair. Prior work on breaking CAPTCHAs states that if the accuracy of the automated CAPTCHA solver is above 0.7%, the particular CAPTCHA service is considered to be broken. We show that we were able to break the state-of-the-art CAPTCHA service by Google, called the 'No CAPTCHA ReCAPTCHA' with 98.4% accuracy. We also present results for seven other CAPTCHA services which too we now consider to be broken.

We consider the following to be our main technical contributions:

- We present the first large scale study of almost all the CAPTCHA systems that offer an alternative audio challenge along with their text-based challenge.

- We build and evaluate a novel, low-cost system to solve the audio challenges of eight different CAPTCHA services using five different solvers. As two of the Speech Recognition services offer transcriptions in 2 different accents, we consider them to be separate solvers.

- Using different accents and acoustic models offered by the speech recognition services, we determine the best method to recognize the digits/characters/words in the challenge to achieve higher efficiency.

3. BACKGROUND

The main idea behind CAPTCHAs is to provide users with a task which only a human can perform, and cannot yet be performed by automated web programs, known as 'bots'. Two-factor authentication is a relatively new mechanism to make sure that the email and phone number of the user is confirmed to authenticate them as real users, although it requires extra effort on the users' part. Text based challenges are still very lucrative and are widely used control mechanisms to prevent automated form filling as they do not require any extra effort in the form of learning, training or access to mobile phones.

Text based challenges evolved over time from being simple fonts embedded in images to cursive, handwritten text. With the evolution of Computer Vision over time, simple challenges proved ineffective in beating bots and the systems were redesigned to use a combination of distorted handwritten text with a lot of noise in the form of background images, skewed lines and using both black and white text. Speech based challenges came soon after their visual counterparts [8, 10] and just like advancements in Computer Vision, there have been similar advancements in natural language processing and in speech-to-text conversion. According to Shirali-Shahreza et al. [18] three groups of people have trouble with this form of CAPTCHA - Visually impaired, who constitute 2.6% of the world's population, users with dyslexia, and users suffering from motor impairment diseases like Parkinson's.

Google introduced reCAPTCHA v2.0 in the year 2014, with the tagline 'Easy on humans, tough on bots'. The new version claimed to use an "advanced risk analysis sytem" that eased users' efforts in solving CAPTCHA challenges. As opposed to the garbled text with random digits and words, this version required the user to click in a checkbox or solve challenges with images. This improved the user experience, while solving the same purpose as the text challenges and was seen as a welcome change by users.

4. ACCESSIBILITY PROBLEMS WITH AUDIO CAPTCHAS

According to Shirali-Shahreza et al. [18] three groups of people have trouble with visual challenges - Visually impaired, who constitute 2.6% of the world's population, users with dyslexia, and users suffering from motor impairment diseases like Parkinson's.

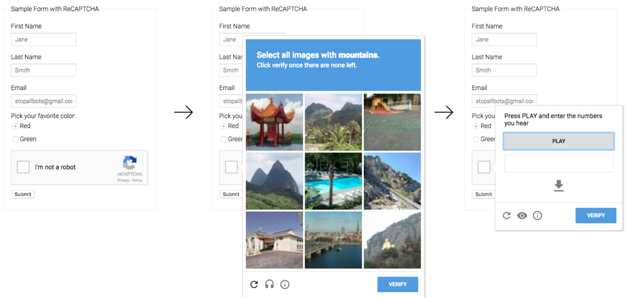
Chellapilla et al. [9] on designing Human-friendly Interaction Proofs (HIPs) argue that HIPs must approach a success rate of at least 90%. Adding to this, they also state that the attack on such a system by automated computer software/scripts must not be successful for more than 1 in 10,000 challenges (0.01%). A study conducted by Sauer et al. [16] which dealt with evaluating audio CAPTCHAs with visually impaired users found that they could only solve the challenges at a rate of 46%, 44% less then what is supposed to be. They also found that the average amount of time taken to correctly solve an audio challenge was 65.64 seconds, which is greater than the 51 seconds that is suggested as the time to complete a CAPTCHA [17].

While conducting a study with blind high school students to inspire them to pursue Computer Science, Bigham et al. found that when the students were presented with an audio CAPTCHA, none of them were able to solve the challenge and their sighted instructors ended up solving the visual version instead [3]. In a subsequent study conducted by Bigham et al with 89 blind users, they found that the users achieved only a 43% success rate in solving 10 popular audio CAPTCHAs [4].

In the same study [4], it was also found that screen readers used by blind users speak over playing CAPTCHAs. As users navigate to the answer box, the accessibility software continue speaking the interface while talking over the playing audio challenge. A playing audio challenge does not pause for solvers as they type their answer and reviewing an audio CAPTCHA is cumbersome, often requiring the user to start again from the beginning. Also, replaying an audio CAPTCHA requires solvers to navigate away from the answer box in order to access the controls of the audio player. Thus, the authors proposed a system and optimized the interface of popular audio CAPTCHA services without altering the underlying implementation and found that the performance increased to 59%.

5. PREVIOUS WORK

Although not as extensively studied as the text versions of CAPTCHAs, there have been several studies in the past which have tested and evaluated of audio challenges. The challenge in breaking an audio CAPTCHA lies in automatically recognizing the sequence of spoken words in the audio while ignoring the noise that is added to the file. There is no universally agreed upon value for the success rate to consider a system broken, although the fact that the trade-off between security and usability needs to be taken into account as well. The study by Chellapilla et al. [9] suggests that the success rate of a bot must be less than 0.01%,



Sample Form with ReCAPTCHA

First Name: Jane

Last Name: Smith

Email: stopallbots@gmail.coı

Pick your favorite color: ● Red ○ Green

I'm not a robot
reCAPTCHA
Privacy - Terms
Submit
Select all images with mountains.
Click verify once there are none left.
VERIFY
Press PLAY and enter the numbers you hear
PLAY
VERIFY

Figure 1: Steps to activate an audio challenge from NoCAPTCHA ReCAPTCHA

whereas Bursztein et al. [5] suggest that the success rate be 1%, and 5% by Tam et al. [12].

The above numbers depend heavily upon many other factors than just being able to successfully transcribing the audio. This implies that the threat model should not take into account just the transcription, but also factors like rate-limiting and the threat also increases with increase in the number of resources that is at the attacker's disposal.

Recent years have seen an increase in the number of studies in breaking audio challenges. It has been shown that audio challenges very prone to attack with ever increasing advancements in the domain of machine learning. As audio challenges consist of a very limited corpus in the form of digits and alphabets and a narrow form of accents, the defense of audio challenges is very less. Existing studies show that most audio CAPTCHAs can be broken by means of machine learning techniques at relatively low costs [5, 7, 12]. Hence, additional defenses in the form of IP address geo location, mouse movement, response times, user agents, cookies, etc. are deployed.

Sivakorn et al. [19] demonstrate 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%. This led us to test the accuracy of audio based CAPTCHA systems from different services. We found that the existing mechanisms in place include rate limiting the number of challenges that are served per IP for every device, including background noise to prevent speech recognizers from recognizing the audio CAPTCHAs automatically. We demonstrate that these limits can be bypassed and that the speech recognizers can be tuned to recognize digits, alphabets and words even with background noise.

The success rates for breaking audio CAPTCHAs reported in recent studies are very high. Bursztein et al. [5] broke the audio challenges of Yahoo, Microsoft, and eBay with a success rate of 45%, 49%, and 83% respectively. Sano et al. [15] and Meutzner et al. [13] broke Google's reCAPTCHA with 52% and 63% accuracy respectively. There have also been studies that have used state-of-the-art automatic speech recognition techniques, based on hidden Markov models (HMMs).

6. SURVEY OF AUDIO CAPTCHA SYSTEMS

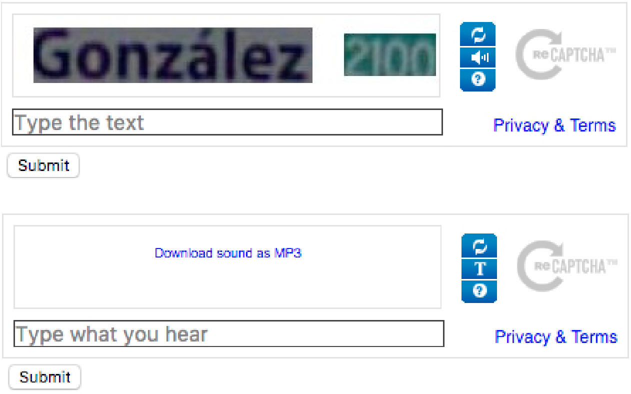


Figure 2: Audio challenge in ReCAPTCHA V1

6.1 ReCAPTCHA by Google (The No CAPTCHA ReCAPTCHA)

Google provides a demo page for ReCAPTCHA, its state-of-the-art CAPTCHA system (the version 2 of ReCAPTCHA), accessible via the following URL (https://www.google.com/recaptcha/api2/demo), for anyone who wants to try out the new CAPTCHA system. We used this demo page for all our tests on ReCAPTCHA. The page connects over HTTPS and does not require a login. The page initially loads a widget with a checkbox, which on clicking, shows another widget with the image CAPTCHA that loads in a new iframe. The audio CAPTCHA challenge appears only after clicking on the headphones-like icon in the bottom of the second widget.

The audio challenge provided by the reCAPTCHA system spells out 5 digits (the system has now been updated with 10 digits) consecutively with little or noise, which the user must then identify and enter into the textbox provided. The interval between the digits is not a constant value. Each digit may be spelled out by the same person or different persons, male or female. Accents do not generally differ to a large extent.

The challenge also provides an option for the user to download the .mp3 version of the audio. The reCAPTCHA system provides a margin for error and allows up to 2 incorrect digits out of the 5. We demonstrate how this high margin for error can be leveraged to get an almost 100% accuracy value. We were able to exploit a security flaw in this system that allowed us to bypass the rate limiting that was in place.

The reCAPTCHA demo page generates a new CAPTCHA challenge each time we connect to it. It behaves exactly like the reCAPTCHA v2 widget that is embedded in popular websites such as Quora, BusinessInsider, Twitter, Ticketmaster and many more, as part of their account creation process. This was one of the main reasons why we chose to do our tests with the demo page, instead of any particular website that employs Google's reCAPTCHA. Our results would thus stay valid on any website that has the latest reCAPTCHA widget embedded.

6.2 ReCAPTCHA by Google (Invisible)

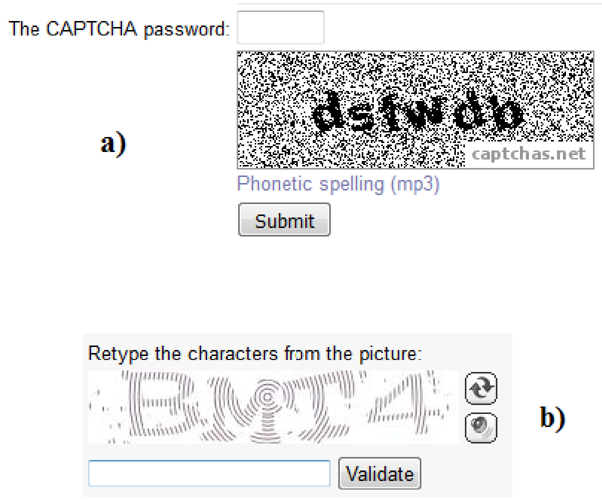


Figure 3: Challenges from a) Captchas.net and b) BotDetect

The latest version of ReCAPTCHA (as of early 2017) is known as the invisible ReCAPTCHA (accessible via the following URL https://www.google.com/recaptcha/api2/demo?invisible=true). This version does not have an explicit checkbox as seen in the previous version. It relies heavily on the number of requests generated per IP address, cookies and other parameters rather than the text/audio challenge itself. If the system does not suspect bot activity, it automatically authenticates the user as real even without presenting the user with a challenge (i.e. without any user interaction). However, a challenge is presented when the system suspects bot activity, and this challenge is very similar to the previous version with respect to the look and feel.

When accessed from a private browsing session, as there are no cookies, the system is suspicious and requires a challenge to be solved. This version spells out 10 digits in its audio challenge consecutively with little or no noise at all, with an error margin of 1 digit and also allows downloading of the MP3 file from the interface itself. Although we were very easily able to transcribe the digits of the audio, we were unable to bypass the rate-limiting that has been set by the system.

6.3 Securimage

Securimage is an open source CAPTCHA system that was developed by Drew Phillips that has been under active development and maintenance since 2005. It also has plugins/modules for major content management systems like Drupal, Wordpress, Joomla!, etc. It is written in PHP and offers very easy integration with websites written in PHP with a lot of customization options for both image and audio based challenges. The demo page contains 3 different forms of challenges - a random digit/character combination, a math puzzle and a custom CAPTCHA made from a word list. The demo page can be found here: https://www.phpcaptcha.org/try-securimage/

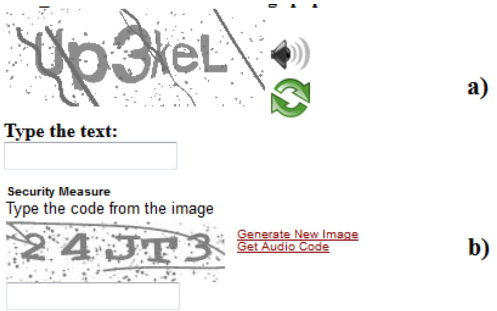


Figure 4: Challenges from a) Securimage and b) RadCAPTCHA by Telerik

For each audio challenge, the system generates a WAV file from a corpus of audio files of 26 alphabets (A-Z) and 10 digits (0-9). The digits and alphabets are spoken by only one person. The audio is generated by picking 6 word/digit files at random, combining them and adding one of the 5 noise files. The default number of digits/characters per challenge is 6, but this can be customized while implementing the library. Some other customization options are that one can specify a word list for the challenges and using a read-aloud math puzzle that is to be solved by the user. Securimage does not offer any margin for error.

6.4 Captchas.net

Captchas.net, started back in 2004 offers implementation and easy integration for PHP, ASP, Perl, Python, JSP and Ruby. It is a free service, but shows a watermark on the images of the text challenges which can be removed by subscribing to the paid version. For both the text and the audio challenges, Captchas.net follows the following method to validate the response:

- Both the client server and Captchas.net's server share a common secret key.
- The secret key and a random string sent by the client are used to generate the password.
- The password is computed by Captchas.net's server to generate the image and by the client server to validate the user's input.

The text challenges always consist only of alphabets and the audio challenge uses NATO alphabets of the same text challenge. For instance, 'RIPAZH' would be read out as 'Romeo, India, Papa, Alpha, Zulu, Hotel' and the user is expected to enter the first letter of each uttered word. There is no margin for error in this service and there is very less background noise.

6.5 ReCAPTCHA by Google (Version 1)

Although still used in many websites, Google has discontinued this version of ReCAPTCHA since 2014. The de facto version used by Google and deployed on most of the websites is the one described in Section 6.1 and 6.2. Signing up for ReCAPTCHA now enables the newer version by default. It must be noted that the websites that used this version previously did not automatically upgrade to the No CAPTCHA ReCAPTCHA version, and they remain open to the vulnerabilities that were fixed in the subsequent version.

There are 10 digits that are spoken in the audio challenge and there is very little or no noise. This version does not enforce any rate limiting unlike the newer versions of ReCAPTCHA. The margin for error is again 1 out of the 10 spoken digits. With respect to the text based challenge, there are two words shown which must be recognized by the user and typed into the input field. The first word is an unknown text from a failed OCR scan while digitizing physical books and other printed material, and the other being a 'control' word, which is known. This helped transcribe the unknown words and digitize them, and in 2012, photographs of house numbers taken from Google's Street View project began to be used, in addition to scanned words.

6.6 Apple

Apple's CAPTCHA is encountered while signing up for a new iCloud account via the following URL: https://appleid.apple.com/account#!&page=create Before spelling out the audio challenge, the voice first says "Please enter the 'n' numbers that you hear in the textbox provided". The spoken numbers consist of two digits like '25', '64', etc. and they are always spoken by a single person. The page times out after two minutes of inactivity and a pop up appears for confirmation if the user is still in the process of creating the account. Apple does not allow downloading the audio file and is loaded and played via JavaScript and not stored in any cache or temporary storage.

To obtain the audio file for the challenge, one has to manually open the URL chrome://media-internals on the Google Chrome browser. The Media Internals feature of Chrome displays three things [1]:

- Everything it can dig up from the media stack about active media players. Includes buffered data, video properties, measured times between events, and a log of events.
- Status and volume of active audio streams. These are not yet associated with a particular tab.
- Cache activity, including reads from and writes to the media cache.

6.7 RadCaptcha by Telerik

RadCaptcha is a CAPTCHA service by Telerik that is built especially for ASP.NET web applications. RadCaptcha and 90+ other controls are part of UI for ASP.NET AJAX, a comprehensive toolset that take care of the common functionality of a ASP based CMS like Sitefinity [2]. It is also a part of a feedback form for the HR website of UIC, where we discovered this CAPTCHA. It can be seen here: https://www.hr.uic.edu/cms/One.aspx?pageId=356957

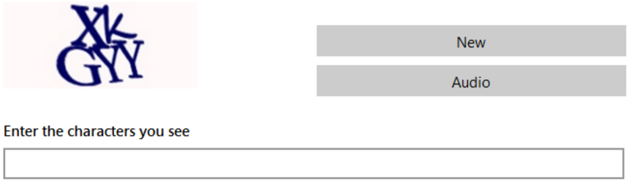


Figure 5: CAPTCHA from Live.com

RadCaptcha's audio challenge speaks out digits (0-9) and alphabets in NATO form. It is spelt out by a single speaker and there is almost no background noise in the audio. The audio is not available to download explicitly, but is present on the DOM of the web page in an <audio> tag, from which the URL of the audio file can be easily grabbed. One of the customizations provided by the service is a rate limit system, which requires the form to be displayed to the user for a particular number of seconds before it can be considered as human-submitted. However, this setting is left to the person who implements the CAPTCHA service on the web page.

6.8 BotDetect

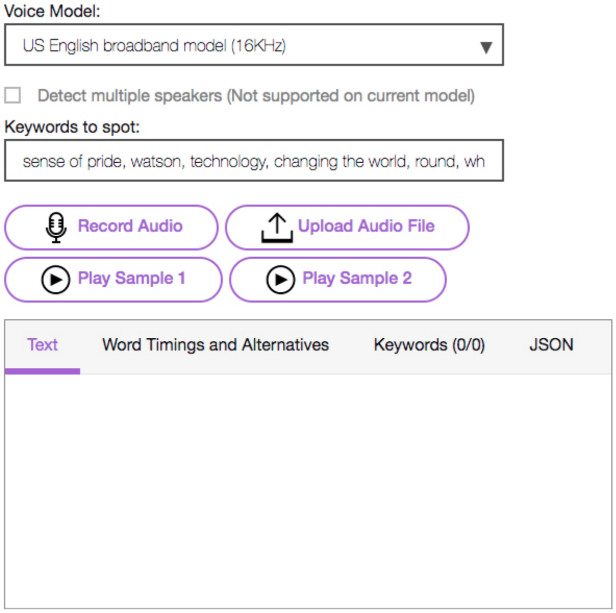
BotDetect offers free and paid versions of its CAPTCHA system. In the free version, the service has several limitations, like the audio challenge is not fully functional most of the times. There is a 50% chance that the audio challenge is replaced with an audio file that speaks "Sound Demo". BotDetect offers libraries for integration with .NET, Java, ASP and PHP based websites. The demo page of BotDetect can be accessed here: https://captcha.com/demos/features/captcha-demo.aspx

The audio challenge speaks out exactly the digits and characters in the text/image challenge. While implementing the CAPTCHA service on one's website, the web master has a lot of customization options. For instance, the number of alphanumeric characters in the challenge can be chosen between 1-13 (the default being a random of 4-6). The background noise can be random or selected from one of the 10 noise types and the sound type can be customized as well (choice of 8 Bit Mono audio or 16 Bit Mono audio). We found that this service is used widely, and we encountered it on the Visa status check page of US CEAC website here: https://ceac.state.gov/ceacstattracker/status.aspx

6.9 Live.com

Live.com, owned by Microsoft is a service that provides access to other Microsoft services. Some of these are Windows, Microsoft Store, OneDrive, Bing, MSN, Office, Skype, Outlook.com, Xbox Live, etc. The CAPTCHA is encountered while registering for a new account with Live.com at https://signup.live.com

The audio challenge of Live.com speaks out random words from the English vocabulary. On an average, they are 5 letter words and 5-7 such words are spoken per challenge, which must be typed by the user. The words are spoken by 2 different people - one female and one male, although we found that the number of words spoken out by the female voice are much higher. In our opinion, it provided better acces-



Voice Model:

US English broadband model (16KHz)

☐ Detect multiple speakers (Not supported on current model)

Keywords to spot:

sense of pride, watson, technology, changing the world, round, wh

Figure 6: IBM Watson's online Speech to Text interface with option to select the Voice Model (accent) and keywords to recognize from the audio.

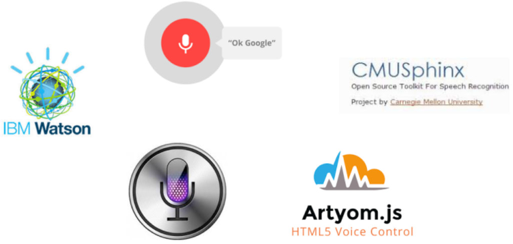
sibility to visually impaired users as they can press '1' on their keyboard to play or repeat the audio challenge and by requesting a new challenge using the button labeled 'New' on the interface, the focus is set to the response box automatically.

7. INITIAL APPROACH

Before we actually began to implement our system, we tried a few approaches to check the feasibility of our approaches. Using a few of the speech recognition services like Google assistant and Siri, we tried playing the audio files of the CAPTCHAs after downloading them. This approach however failed as the systems were unable to recognize the audio. Next, we tried the possibility of using a mobile application that does specialized speech recognition and it was able to recognize about 2-3 digits from the audio when it was played under a normal room environment. This gave us the motivation that such dedicated speech recognizer would work.

Next, we tried playing the audio from the system while recording it via the microphone at the same time. This had a very poor accuracy, but it led us to the next step, where we rerouted the system audio back to the microphone to mimic the computer talking to itself. This too provided a very poor accuracy. We realized that the real time continuous speech recognition libraries were poor in accuracy in recognizing the audio CAPTCHAs. This led us to explore speech recognition from audio files.

Our next approach was to manually upload these files to speech recognition service that transcribed audio files. We used the IBM Watson's Speech to Text service and were able to transcribe the audio file. The accuracy was not much, but it did provide a few alternative transcriptions, which had the correct values. This indicated that our attack was feasible by using these Speech to Text converters.



IBM Watson
"Ok Google"
CMUSphinx
Open Source Toolkit For Speech Recognition
Project by Carnegie Mellon University
Artyom.js
HTML5 Voice Control

Figure 7: A few speech recognition services.

8. OFF-THE-SHELF SPEECH RECOGNITION SERVICES

For the various solvers that we built, we used three speech recognition services - IBM Watson's Text to Speech, Wit.ai by Facebook, and Google Speech API. We call these off-the-shelf speech recognition as we did not have to build a speech recognition system using deep learning or NLP on our own to solve the audio challenges.

When we started off with IBM Watson's Speech to Text API, we found that it had two accents to choose from. We also found that transcribing the same audio file using the two accents fetched us different results. In a similar case with Google's Cloud Speech API, we found we could choose from various accents in English - United States, United Kingdom, Australia and India to name a few. Thus, when we used the IBM Watson and Google Cloud Speech API solvers, we decided to explore two different accents - US and UK - and evaluated the results that we obtained by transcribing the audio with each. Wit.ai does not provide an option to specify the accent in which the audio is to be recognized.

We found that both IBM Watson and Google Speech API solvers provide a list of alternative transcriptions for each recognized token in the audio. While building our response to the challenge based on the received transcription, we also looked for the tokens we wanted (like digits and alphabets) among the alternative transcription. This improved our accuracy to a great extent than just using the direct transcription received from these services. The results from these solvers were processed to convert words to digits ("one" as "1", etc.).

IBM Watson service uses "machine intelligence to combine information about grammar and language structure with knowledge of the composition of an audio signal to generate an accurate transcription". It is available for free for the first 1000 min/per month for each account, after which each additional minute is charged $ 0.02. We used multiple accounts each to access the free version of the software for the term of our project. This service can either be accessed through a WebSocket connection or through its REST API. The website provides sample code in Curl, Ruby and Node.JS with best practices for using their REST API. The API is also

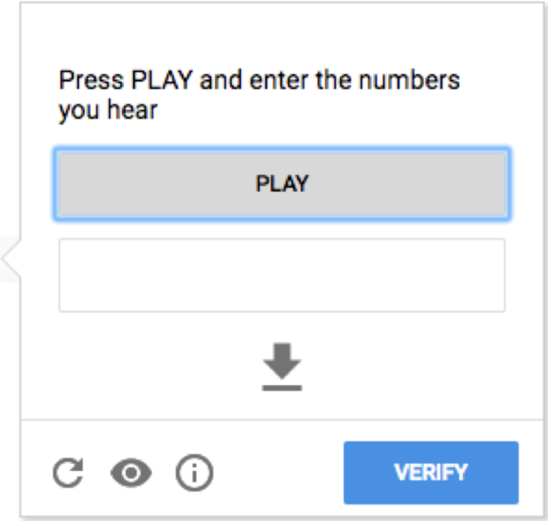

Press PLAY and enter the numbers you hear

Figure 8: The ReCAPTCHA widget: Users can play the audio, download the file or request for a new challenge.

very well documented.

9. SYSTEM OVERVIEW

9.1 Generic System

Our system consists of a crawler that loads the required page, finds the audio element in the page and downloads it. It then converts the audio file into the required audio format and hands it over to another system that connects to the speech recognition web services. Once the transcript is obtained from the service, it hands over the result to the first system that fills the CAPTCHA response and submits the form/challenge.

To build the crawler, we used a Python script with Selenium and ChromeDriver. The script recursively loads the URL that has been hard coded into the system and with Selenium, we fire up the ChromeDriver. Selenium helps in finding the required elements on the page and manipulate them. Once it finds the audio element on the page, it is downloaded and if the file is in MP3 format, we use the Pydub library to convert the file to WAV/FLAC format. We did this as IBM Watson's Speech to Text API expects only a WAV file and Google Speech API expects a FLAC file. These two files are stored with the timestamp of when the first file was initially downloaded.

Next, the control is transferred to the solver script. We implemented this in Node.JS and we used the Naked library available for Python to execute a JavaScript file. The Node.JS script is fired up that takes the converted file and sends it to the appropriate solver using its API and gets back the response. In the case when the solver gives a list of alternative transcriptions for each recognized sound, we select the most appropriate one among them.

We did not require transferring control to the Node.JS module for the Google solvers as we used the Python implementations of Google Speech API.To achieve this, we used the Google Speech API Client module that is available to install via the "pip" command for Python and setting up the credentials via a JSON file that can be downloaded by registering for a new application with the Google Cloud platform.

The script then builds the response based on this and sends it back to the Python script. It then enters the response into the response box using Selenium, and the challenge is submitted. Once it is submitted, we look for some element on the page that is generated which tells the user if the submitted response was correct or not. We again crawl the page for this element and gather this result. We next store these results in a CSV file along with the file name for each challenge we solve.

9.2 Tweaking for ReCAPTCHA

We found that ReCAPTCHA had implemented rate limitation (we talk about this in detail in the forthcoming section). To overcome this limitation, we had to tweak our script to generate random user agent strings that would allow us to bypass the limit. It also generated the audio in MP3 format that had to be converted into WAV.

9.3 Tweaking for Captchas.net and Telerik

Captchas.net and Telerik CAPTCHAs spell out NATO characters and expect the users to solve the challenge using English alphabets. To achieve this, we had to modify our script to map the NATO responses to English alphabets.

9.4 Tweaking for Live.com

As we would have created spam accounts on Live.com in order to verify to evaluate our attack, we preferred not to flood their system with fake accounts. Thus, we modified our system to only crawl and download 100 audio challenges that we manually transcribed and then used the 5 solvers on that dataset to verify with the truth set that we had created.

9.5 Tweaking for Apple

As we did not get the audio file from Apple's CAPTCHA directly, we had to modify our script to open the chrome://media-internals URL simultaneously to grab the playing audio. We also had to send null values to the response box and vary the time delay between the various input fields to mimic a human.

Listing 1: Setting Expected Phrases in Google Speech API for Captchas.net solver

```
’speechContext’: {

        ”phrases”: [”Alfa”, ”Bravo”, ”
            ↪ Charlie”, ”Delta”, ”Echo”, ”
        ↪ Foxtrot", "Golf", "Hotel", "
        ↪ India", "Juliett", "Kilo", "
        ↪ Lima", "Mike", "November", "
        ↪ Oscar", "Papa", "Quebec", "
        ↪ Romeo", "Sierra", "Tango", "
        ↪ Uniform", "Victor", "Whiskey
        ↪ ", "X-ray", "Yankee", "Zulu"
    ]

}
```

10. EXPERIMENTAL EVALUATION

In this section, we evaluate our automated system against all the previously mentioned CAPTCHAs. We launch an attack on the demo or live pages of these services, grab the audio element and transcribe it with our automated solver and record the results.

10.1 Parameters for our attack

We tried testing our solvers on a minimum of 50 audio CAPTCHAs to get started and discover the best solver for each service. The first service which we went for was the No CAPTCHA ReCAPTCHA, which we found to be the hardest. We tuned our automated solver to work for an iteration of i<=100 and ran it against the same reCAPTCHA demo page using IBM Watson US solver.

While testing for i<=100, we hit an interesting roadblock. Our system was performing very well till the first seven iterations and after the seventh iteration, we observed that all our challenges failed. After close examination, we found that Google's reCAPTCHA system was blocking requests from us after the seventh try. It was returning an audio file with the message saying "We're sorry. Your network or computer might be sending us automated queries. To protect our users, we can't process your request right now." So our system which was trying to convert words to digits returned all zeroes, indicating that the challenge could not be solved.

We tried various things to get past this defense implemented by Google for Denial-Of-Service attack (DOS) attacks. The following is the list of avenues that we explored, but failed:

- **Incognito:** Opened the driver in incognito mode, so that the server did not associate the requests from our system to our user cookies.
- **Cache:** Cleared the cache in the driver's browser to remove any temporary files that might have been stored from the last time we visited the URL.
- **Session:** Set the *ensureCleanSession* variable to true to remove all session information, even though we did not log in.
- **Window size:** Changed the window size for each iteration, to fool the system into thinking that a human might be using the system.
- **User Agent:** Gave a valid User Agent string, to make sure that an invalid entry does not make us look any suspicious.
- **Tor:** Crawled the page using the selenium browser extension on Tor, to mask the IP address. But could not get to the iFrame containing the Audio challenge, because of a difference in the DOM structure in Firefox, which Selenium was not able to detect.

Listing 2: Setting up arguments for ChromeDriver in Python using Selenium

```
capabilities = DesiredCapabilities.
    ↪ CHROME
capabilities['ensureCleanSession'] =
    ↪ True
options = webdriver.ChromeOptions()
options.add_argument("-incognito")
options.add_argument("-disable-cache")
options.add_argument("'chrome.prefs': {'
    ↪ profile.
    ↪ managed_default_content_settings.
    ↪ images': 2}")
options.add_argument('--window-size
    ↪ =1903,719')
options.add_argument(
    '--user-agent' + "Mozilla/5.0 (Linux;
        ↪  Android 6.0.1; SM-G920V Build
        ↪ /MMB29K) AppleWebKit/537.36 (
        ↪ KHTML, like Gecko) Chrome
        ↪ /52.0.2743.98 Mobile Safari
        ↪ /537.36")
options.add_experimental_option("
    ↪ excludeSwitches", ["ignore-
    ↪ certificate-errors"])
options.add_argument('--clear-token-
    ↪ service')
```

We tried all the above settings individually and together, but all to no avail (Listing 2). After all these failed attempts, two different approaches worked for us and helped us get past the 7 CAPTCHAs/day/IP/cookie limit.

- **Web Proxy services:** We looked for free web proxies on a website - www.proxylist.hidemyass.com. We filtered the list to select the fastest ones and with the highest anonymity level. A web proxy server acts as a gateway or tunnel, forwarding requests and responses unmodified. We included the IP address of the web proxy server in the driver settings, so that our IP would be masked.

  With this minor modification, we could get more than 100 CAPTCHAs per day. But one limitation with this approach, as we found with our experiments, was that the response time depended on the availability of the proxy server to serve our requests and its location. Since the reCAPTCHA system only had a 2-minute time window and our Python scripts were programmed to wait for certain periods of time, late responses resulted in a *TimeoutException*. This resulted in a very low accuracy rate.

We made two interesting observations from these experiments. One was that the location of the web proxy server and hence the Origin of the request, did not affect the accent of the audio challenges. We tried using web proxy servers from US, UK, Russia, Canada, Mexico, South Africa, Korea, China, Hong Kong and India. The reCAPTCHA system returned similar audio files for requests from all these web proxy services.

Another observation that was made was the requests from US proxy services were trusted more than the ones in other countries. Also, the requests got served faster when compared to requests from other countries.

- **Unique User Agents:** We wanted a better solution so that the accuracy of our solver could be determined properly. We generated a completely unique and gibberish User Agent string to fool the defense system that the requests were not coming from the same user (Listing 3). Surprisingly, the reCAPTCHA DOS defense system does not check any of the other fields - IP, cookies - User/HTTP, legitimate user behavior when it encounters a User Agent string that it was not expecting. This is a bug in the implementation of the defense system in ReCAPTCHA that we discovered.

  Listing 3: Generating unique User Agents with current time

  ```
  timestr = datetime.datetime.now().
      ↪ strftime("%Y%m%d-%H%M%S")
  options.add_argument("user-agent="+
      ↪ timestr+"lol")
  ```

  We leveraged this bug to attain and solve more than 1000 CAPTCHA challenges per day, even from the same IP address.

Once we exploited this vulnerability in ReCAPTCHA (which as now been fixed), we used the same parameters for all the other services had did not encounter any rate limiting problems (Except for Apple, which we described earlier).

Listing 4: Returned response from Google Speech Api for an acoustic captcha sample

```
{"results": [{"alternatives": [{"transcript
    ↪ ": "eight", "confidence":
    ↪ 0.982679}]}, {"alternatives": [{"
    ↪ transcript": " three", "confidence":
    ↪  0.7421026}]}, {"alternatives": [{"
    ↪ transcript": " eight", "confidence":
    ↪  0.59319055}]}
, {"alternatives": [{"transcript": " nine",
    ↪  "confidence": 0.6004673}]}, {"
    ↪ alternatives": [{"transcript": "
    ↪ five", "confidence": 0.982679}]}]}
```

10.2 Improving Accuracy

To further improve our accuracy rates, we looked at a feature that was available in both IBM Watson's Speech to Text and Google Speech API. The converters could be configured to look for a specific set of words or a dictionary, which improved our accuracy. For the ReCAPTCHA system, we configured the converters to look for the 10 digits, including zero. For Captchas.net, we added a dictionary of all the 26 NATO phonetic alphabets.

Also, we tried to customize our solvers specifically for each of these CAPTCHA services. We looked for words that sounded similar to digits in case of the reCAPTCHA system and the NATO alphabets in case of Securimage. We manually listened to over 50 challenges from each service to build a corpus of the common words that were misidentified. This greatly increased the accuracy of our solvers. Noise reduction would be a good way to further increase the accuracy.

11. RESULTS

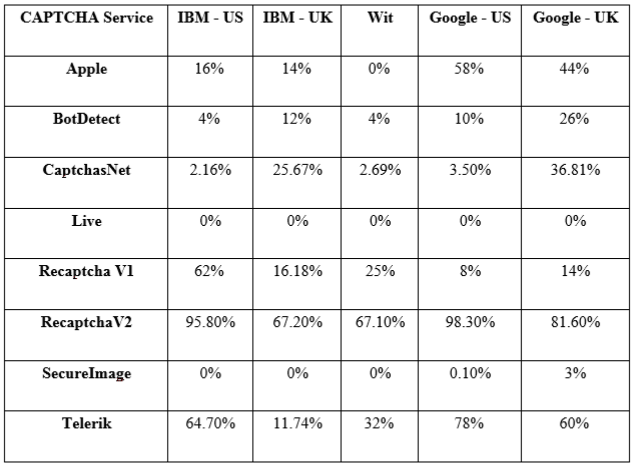
We demonstrate that our attacks are highly effective against the state-of-the-art ReCAPTCHA service by Google. Together with the user agent vulnerability that we uncovered and using Google's Cloud Speech API, our attacks are most effective against Google's reCAPTCHA than other existing CAPTCHA providers as shown in Fig.3. This high accuracy was based on little or no background noise in ReCAPTCHA's audio and highly efficient speech recognition systems. Although the vulnerability has since been fixed, the audio challenges remain very simple to crack.

We found that the US accent solved the challenges with a higher accuracy than the UK accent, with the exception of Securimage and Captchas.net where we had higher accuracy with the UK accent solver from Google's Cloud Speech API. We also found that IBM Watson's Speech to Text API was the next best solver for ReCAPTCHA's audio challenges. 0The increase in time for the other services was due to the background noise and more complex challenges than ReCAPTCHA.

We also show that our attack was least effective against Securimage, which uses background noise to mask the audio. We would like to mention here that Securimage is an open source system that provides the source files, including the noise files which would allow the attacker to easily filter out the noise as it is known beforehand. We consider this to be an important future work.

12. FUTURE WORK

Our system relies on off the shelf tools that did not respond to reduction of noise using simplistic techniques such as applying sox commands to remove certain frequencies. Noise removal is necessary for improving accuracy for Securimage, BotDetect and all other services. Our future work will focus on generating noise profile for Securimage using tools like MATLAB or Audacity to filter out noise and using



CAPTCHA Service	IBM - US	IBM - UK	Wit	Google - US	Google - UK
Apple	16%	14%	0%	58%	44%
BotDetect	4%	12%	4%	10%	26%
CaptchasNet	2.16%	25.67%	2.69%	3.50%	36.81%
Live	0%	0%	0%	0%	0%
Recaptcha V1	62%	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%	78%	60%

Figure 9: Results: Accuracy of solving various CAPTCHAs using the different solvers.

a deep learning system.

Our work shows that in an effort to improve the CAPTCHAs for human understanding they have become vulnerable to off the shelf ASRs. Further research is required to improve the functionality of CAPTCHA's as their intended use to prevent automated solvers such as ours gaining access to content.

13. CONCLUSION

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 did not train our system using AI and Machine Learning. We evaluated the various CAPTCHA providers against the various solvers that we used and presented the results along with accuracy and the time taken to solve the challenges. We also discussed how we could improve our system in the future to increase our attack against the services we evaluated.

14. REFERENCES

[1] The chromium projects: Media internals.

[2] Radcaptcha - telerik asp.net captcha.

[3] J. P. Bigham, M. B. Aller, J. T. Brudvik, J. O. Leung, L. A. Yazzolino, and R. E. Ladner. Inspiring blind high school students to pursue computer science with instant messaging chatbots. In *ACM SIGCSE Bulletin*, volume 40, pages 449–453. ACM, 2008.

[4] J. P. Bigham and A. C. Cavender. Evaluating existing audio captchas and an interface optimized for non-visual use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1829–1838. ACM, 2009.

[5] E. Bursztein, R. Beauxis, H. Paskov, D. Perito, C. Fabry, and J. Mitchell. The failure of noise-based non-continuous audio captchas. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 19–31. IEEE, 2011.

[6] E. Bursztein and S. Bethard. Decaptcha: Breaking 75% of ebay audio captchas.

[7] E. Bursztein and S. Bethard. Decaptcha: breaking 75% of ebay audio captchas. In *Proceedings of the 3rd USENIX conference on Offensive technologies*, page 8. USENIX Association, 2009.

[8] T.-Y. Chan. Using a test-to-speech synthesizer to generate a reverse turing test. In *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on*, pages 226–232. IEEE, 2003.

[9] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski. Designing human friendly human interaction proofs (hips). In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 711–720. ACM, 2005.

[10] G. Kochanski, D. P. Lopresti, and C. Shih. A reverse turing test using speech. In *INTERSPEECH*, 2002.

[11] A. L. Maas, V. L. Quoc, and T. M. O'Neil. Recurrent neural networks for noise reduction in robust asr.

[12] H. Meutzner, S. Gupta, V.-H. Nguyen, T. Holz, and D. Kolossa. Toward improved audio captchas based on auditory perception and language understanding. *ACM Transactions on Privacy and Security (TOPS)*, 19(4):10, 2016.

[13] H. Meutzner, V.-H. Nguyen, T. Holz, and D. Kolossa. Using automatic speech recognition for attacking acoustic captchas: The trade-off between usability and security. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 276–285. ACM, 2014.

[14] D. Networks. Distil networks study reveals captchas have negative impact on web traffic and leads.

[15] S. Sano, T. Otsuka, and H. G. Okuno. Solving google's continuous audio captcha with hmm-based automatic speech recognition. In *International Workshop on Security*, pages 36–52. Springer, 2013.

[16] G. Sauer, H. Hochheiser, J. Feng, and J. Lazar. Towards a universally usable captcha. In *Proceedings of the 4th Symposium on Usable Privacy and Security*, volume 6, page 1, 2008.

[17] T. Schluessler, S. Goglin, and E. Johnson. Is a bot at the controls?: Detecting input data attacks. In *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, pages 1–6. ACM, 2007.

[18] S. Shirali-Shahreza and M. H. Shirali-Shahreza. Accessibility of captcha methods. In *Proceedings of the 4th ACM workshop on security and artificial intelligence*, pages 109–110. ACM, 2011.

[19] S. Sivakorn, I. Polakis, and A. D. Keromytis. I am robot:(deep) learning to break semantic image captchas. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 388–403. IEEE, 2016.