Decaptcha: Breaking 75% of eBay Audio CAPTCHAs

Elie Bursztein
elie@cs.stanford.edu

Steven Bethard
bethard@stanford.edu

Abstract

CAPTCHA tests aim at preventing attackers from performing automatic registration. In this paper we show that our prototype *Decaptcha* is able to successfully break 75% of eBay audio captchas. We compare its performance with the state of the art, readily available speech recognition system *Sphinx* and discuss the implications for eBay security.

1 Introduction

Distinguishing computers from humans has become a central issue for website security, as many services rely on this distinction to work properly. Website registration is a prominent example of such a service: applications must ensure that attackers can't perform website registration automatically. Gmail must distinguish humans from computers to prevent abuse by spammers, and eBay must do so to avoid a flood of scams and illegal items.

Completely Automated Public Turing tests to tell Computers and Humans Apart (CAPTCHA[1]) [12] were developed to allow websites to make this distinction automatically. While the idea behind these tests is simple – find something which is easy for humans and hard for computers – its implementation has proven difficult. The problem of finding efficient captchas has become so prominent that even the New York Times had a column on the subject recently [5].

The security of image-based captchas has received a large amount of attention [1], yet audio captchas, which are often provided alongside image-based captchas to improve website accessibility, have been less explored. To our knowledge, the only academic study [15] to date of audio captcha security appeared in February 2008 and covered only captchas from *Google*, *Digg* and the old *reCAPTCHA* system. At the same time, Wintercore demonstrated a prototype that was able to break Google audio captchas using a similar approach [14].

Motivation. Following the observation that audio captcha security is often overlooked, we evaluate if, a year after the last study, a knowledgeable attacker is able to abuse the registration process of the popular site eBay by breaking the audio captchas. We choose eBay because to the best of our knowledge the security of their audio captchas has never been evaluated. More precisely we wanted to evaluate the following:

- **Feasibility**: Is it possible to abuse the eBay registration system by breaking their audio captcha system? How many captchas per day could an attacker solve?
- **Audio Processing Requirements**: What kind of knowledge and resources are required to break eBay audio captchas? This is an central question as it allows us to understand what kind of attackers would be able to effectively conduct such an attack. This answer can also serve as a baseline to design better captchas and implementation guidelines.
- **Scraping Requirements**: How difficult is the act of automatically collecting a large number of audio captchas? The security of the captcha defense is heavily dependent of its implementation. Anti-scraping mechanisms can play an important role in mitigating captcha attacks by slowing down the attacker. We were interested in analyzing the effectiveness of various choices made by eBay in this domain. Surprisingly it turns out that the eBay captcha implementation is flawed in several ways.

Contribution. Our main contribution is the first attack on the eBay audio captcha system that is able to break **75%** of its audio captchas. To perform this attack we have:

- **Built a scraper**. We have developed a multi-threaded scraper to build our captcha corpus. We scraped more than 26,000 audio captchas to perform our analysis.

[1]For lisbility purpose we will write captcha instead of CAPTCHA in the rest of this paper

- **Evaluated state of art software**. We have evaluated how well a state of art speech recognition system, Sphinx [17], is able to break audio captchas. Our analysis shows that while it can break about 1% of captchas (exceeding the current standard of 0.1% [9]), its accuracy is not high enough to consider it a clear winner.
- **Developped a breaker**. Following the approach of [14] and using an eBay implementation flaw to build a corpus, we have developed our own tool called Decaptcha that is able to break 75% of eBay audio captchas. Combined with a medium sized IP pool, Decaptcha could register over 75,000 fake accounts on eBay per day.

Outline. The remainder of the paper is organized as follows: In Sec. 2 we explain the general methodology for breaking captchas and present our threat model. In Sec 3 we present how we scraped audio captchas from eBay and why their implementation is flawed. In Sec 4 we analyze how eBay audio captchas can be broken and determine how many registrations per day an attacker can achieve. In Sec 5 we discuss the trade-offs that eBay can implement to mitigate captcha attacks. In Sec 6 we provide additional related work. In Sec 7 we conclude.

2 Background

In this section, we summarize how to attack the eBay registration process, explain how to evaluate the performance of a captcha solver and discuss our threat-model.

Breaking the Registration Process. Breaking any registration process involves three steps: **scraping**, **solving** and **registering**. For the **scraping** phase, the attacker develops a program to automatically download the registration page(s) and its captcha. This is the first line of defense: ultimately the attacker is bound by the number of registration pages he is able to fetch each day. Limiting the number of registration attempts per IP can slow an attacker, though one with a large IP pool may still be able to break many captchas. The next phase, **solving**, involves determining the correct answer to the captcha from the registration page. This is the central defense against automatic registration as captchas are designed to be difficult to solve efficiently by computer. Finally in the **registering** phase the attacker fills in the form and submits it. This phase is typically the easiest, and we do not discuss it here – we assume that an attacker who can obtain a registration page and solve its captcha can also register successfully.

Breaking Captchas. The **solving** phase consists of three steps: **preprocessing**, **segmentation** and **classification**. In the **preprocessing** step, parts of the captcha irrelevant to the task are removed. For audio captchas, this can include noise reduction techniques like spectral subtraction. In the **segmentation** step, the captcha is broken into pieces to be individually recognized. A simple audio analysis program might try to break the captcha into digits, while a speech recognition system might break the audio into phones. State of the art of captcha design suggests that segmentation should be the most difficult stage: for example the robustness of text-based schemes (images) relies on the difficulty of finding *where* the character is, rather than on *which* character it is [9, 18]. Finally, in the **classification** step, each identified segment is assigned a label, e.g. a digit or letter. This may be performed by applying a machine learning classifier, or, as in speech recognition, by search. At the end of classification, the system's final prediction is created by combining the individual classifications made for each segment.

Performance Evaluation. The most basic measure of captcha solving efficiency is **accuracy**, which is the fraction of the total captchas that were answered correctly. However, tools may also choose to respond to some captchas and reject others without answering them. For example, since eBay captchas are always 6 digits long, an answer that contains more or fewer digits has clearly been segmented incorrectly. Therefore we also evaluate solvers using coverage and precision. **Coverage** is the fraction of captchas that the tool attempts to answer. In the eBay case, it is the number of catpchas that are correctly segmented into 6 digits. **Precision** is the fraction of the captchas attempted that the solver guesses correctly. Note that current captcha design goal is to ensure that "*automatic scripts should not be more successful than 1 in 10,000*" attempts (i.e. a precision of 0.01%) [9].

Threat Model. To evaluate the security of a captcha system, the attacker resources (IPs and knowledge) need to be considered. To model various access to these resources, we consider the following four classes of attackers:

- **Low**: This attacker has little knowledge of speech recognition and therefore uses an off-the-shelf sys-

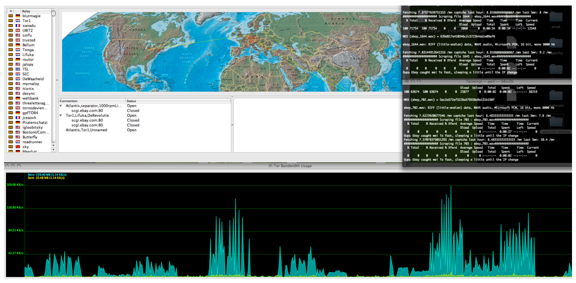

Figure 1: Our scraper in action.

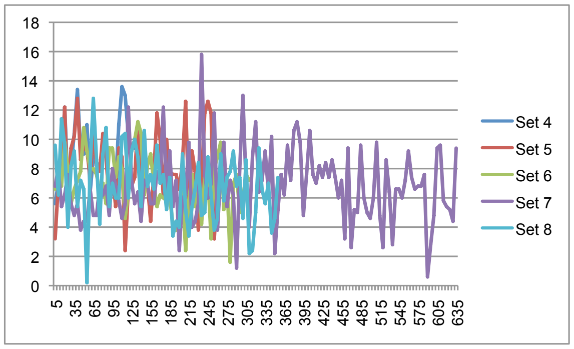

Figure 2: Number of captchas fetched per minute for various set. Statistics aggregated in 5 min intervals

tem. This attacker uses TOR [4] and accordingly has access to approximately 1500 IPs.

- **Medium**: This attacker also uses off-the-shelf speech recognition, but has access to a small botnet providing around 10,000 IPs [19].
- **High**: This attacker has knowledge of speech recognition, can build a custom tool, and has access to a 10,000 IP botnet pool.
- **Advanced**: This attacker has the kind of advanced knowledge and resources available to the owner(s) of Storm [6] or Conficker [13]. This attacker uses a custom tool and has access to at least 94,335 IPs [11].

3 Scraping

Building a scraper for eBay registration pages served two purposes: First, it allowed us to evaluate how many registration pages and captchas could be pulled each day. Second, these pulled captchas became the training and evaluation data for our captcha solvers.

Scraping Defense. Scraping eBay uncovered two interesting facts. First, eBay allows the same captcha to be re-downloaded but with different voices and noise levels. This allowed us to build a large training corpus with minimal annotation, and therefore to train *Decaptcha* (Sec 4) with minimal human labor. The second fact is that eBay fetching limits are a bit sketchy. You can only fetch about 20 to 40 captchas per minute by IP, then you have to wait for a variable amount of time: in some cases the system asked us to wait 24 hours and sometimes we were able to reuse the same IP after few minutes. Therefore for the rest of the paper we take a conservative approach: we assume that the attacker is not able to fetch more than N captchas per day [2].

Scraping Speed. Our scraper is written in Perl and is able to run multiple threads in parallel. It can run in two modes: Tor mode and botnet mode. The Tor mode uses Tor as a relay for IPs and the botnet mode uses our own IPs as relays. We have implemented a statistical module to optimize the scraping parameters such as the number of captchas to scrape before changing IP or captcha ID, and to evaluate the scraping speed of the two approaches. While using our own IPs provides a stable scraping speed of 60 captchas per minute, the Tor mode speed varies greatly (Fig 2) from 2 to 16 captchas per minute.

4 Breaking CAPTCHAs

In this section we explore how Sphinx and Decpatcha can be used to solve eBay audio captchas.

eBay audio captchas are composed of six digits from zero to nine. Each digit typically comes from a different recording and these are concatenated to produce the audio that the user hears. The different recordings have different noise levels, speaker accents and speaker genders. We observed that some recordings appear to have been programmatically modified by changing the speed of the utterance.

As discussed in the previous section, we have collected a large corpus of eBay captchas: we collected about 2000 samples of eight captchas, and 200 samples for 50 other captchas. We reserved 200 samples of the first eight captchas and all 200 samples of the last 50 captchas to evaluate the performance of the solvers. Overall we scraped more than 26,000 captchas.

[2]We hope that eBay will enforce this kind of policy in the future after realizing that the current system is not secure.

Model	Acc	Prec	Cov
Sphinx TIDIGITS	0.3	9.6	3.6
Sphinx HUB4	1.0	28.9	3.6
Decap 3x100	53.4	65.7	81.3
Decap 3x500	59.6	73.3	81.3
Decap 6x100	65.6	80.7	81.3
Decap 8x1000	75.1	92.3	81.3

Table 1: Performance of the captcha solvers on the 10,000 items in the 50x200 test data. Acc(uracy), Prec(ision) and Cov(erage) are shown for each model.

Breaking captchas with Sphinx.. Sphinx [17] is a state-of-the-art, open source speech recognizer which can be applied to audio captchas. Sphinx handles the usual speech recognition steps: converting audio to features, generating a search graph, and decoding the features and graph to predict the utterance.

To generate the search graph, Sphinx requires both an acoustic model and a language model. Numerous models are freely available for Sphinx, including the TIDIGITS model, tuned to digit recognition, and the HUB4 model, tuned to large vocabulary tasks. Language models are also available for Sphinx, but since eBay captchas are always a sequence of six digits, we used as our language model a simple grammar that required exactly six digits in a row, with optional silence before and after each digit.

Table 1 shows the performance of the Sphinx recognizer with the TIDIGITS and HUB4 models. In both cases, Sphinx produced output for only 3.6% of the captchas it was presented[3]. When Sphinx did produce a transcription for a captcha, it was right only 9.6% of the time with the TIDIGITS model, and 28.9% of the time with the HUB4 model. This is interesting because a naive attacker might expect the digit-tuned model to perform better on the eBay digit recognition task. However, the large vocabulary model performs better, most likely because of its greater exposure to a wider variety of background noise and word pronunciations.

Overall, the best Sphinx model provides only a weak attack capability – it would produce only 10 successful registrations for 1000 downloaded captchas.

Breaking captchas with Decaptcha.. Since off-the-shelf software performs poorly on eBay captchas, we decided to implement our own tool: *Decaptcha*[4]. Following the work of [8] and [15], *Decaptcha* works by looking at voice energy spikes.To do so it applies a discreet Fourier transform (DFT) to the wave file and then isolates the energy spikes. Decaptcha uses a supervised learning algorithm that looks at these decompositions to build the model which it uses to recognize digits. As explained in the background section (Sec. 2), the key difficulty was the segmentation. To be efficient this phase requires a lot of tuning that mainly involves determining three key parameters: the window size, the energy level threshold and the number of bins for the DFT.

The window size defines the size of each audio block, or *sample*. For instance, since eBay captchas have a 8000hz sampling, having 256 samples requires a 128 ms window size. This window size is a trade-off between capturing significant spikes and narrowing them sufficiently so they are discriminatory. The second parameter is the energy level threshold below which the sample is considered noise. This is essential for decomposing the wav into digits and noise. The last parameter is the number of bins that are used to store the frequencies resulting from the DFT. Other parameters come into play such as word overlap and file offset but they play a less significant role than the aforementioned ones.

We tuned Decaptcha parameters on our development set, selecting the set of parameters in which the largest number of samples were successfully decomposed into exactly six parts. The Decaptcha algorithm is fairly sensitive to the various parameters. For example, on a 300 captcha test set with its optimal settings Decaptcha decomposes *254/300* captchas correctly, while with settings only slightly off of these, it decomposes only *124/300* captchas correctly. We could probably improve the performance by applying better pre-processing noise reduction algorithms but given the current accuracy without it, this was not necessary.

We trained Decaptcha on four different sets of data to construct four distinct models from which we could evaluate how many captchas and samples were needed to achieve good accuracy. The first model, **3x100**, uses 100 samples of three captchas that were selected to ensure that each of the 10 digits appears in at least one of the three captchas. The second model, **3x500**, uses 500 samples of the same three captchas. The third model, **6x100**, uses 100 samples

[3]Sphinx can fail to produce output for a number of reasons, but essentially this means no word sequence it searched through was scored highly enough.

[4]Due to the high accuracy of Decaptcha we can't disclose it, however the reader is invited to verify our claims by testing an online version available here: `http://www.dontrythisathome.com/decaptha`

Model	Acc	Prec	Cov
Decap 10x50	44.0%	55%	80%
Decap 25x20	**76.0%**	**95%**	80%
Decap 50x10	72.0%	90%	80%

Table 2: Performance of the 500-sample models on 25 new captchas.

N	Sphinx	Sphinx-d	Decap	Decap-d
2	2.0	2.1	88.2	87.7
5	4.7	4.8	98.8	98.9
10	8.5	8.8	99.9	100.0
50	25.4	27.7	100.0	100.0
100	35.6	43.1	100.0	100.0

Table 3: Model accuracies when considering N additional samples of each captcha. The Sphinx and Decaptcha models are the best ones from Table 1.

of six different captchas, with each digit appearing at least two times. The fourth model, **8x1000**, uses 1000 samples of eight captchas, with each digit appearing at least three times.

Table 1 shows the performance of these four models. All models produce some response for 81.3% of the captchas[5]. The smallest model, trained on only 600 samples, gets 65.7% of these captchas correct, while the largest model, trained on 8000 captchas, gets **92.3%** of them right. Interestingly, the 6x100 model outperforms the 3x500 model (59.6% to 65.6%) even though the latter model was trained with two and a half times more data.

Based on this observation, we hypothesized that a training set with a few samples of many captchas would be better than a training set with many samples of few captchas. To verify this, we trained Decaptcha on three additional training sets: 10 captchas with 50 samples each, 25 captchas with 20 samples each, and 50 captchas with 10 samples each. These models were then evaluated on 25 new captchas collected for this purpose. Table 2 shows that accuracy and precision seem to peak with around 20 samples of 25 captchas. This suggests that we can get away with annotating only 25 captchas, as long as we collect at least 20 variants of each.

Overall, Decaptcha provides a strong attack capability – it would produce 750 successful registrations for 1000 downloaded captchas.

[5]The models have identical coverage because the decomposition depends not on the training captchas but on the manually set parameters which were the same for all models.

Threat	Model	IPs	Reg/day
Low	Sphinx	1,500	150
Medium	Sphinx	10,000	1,000
High	Decaptcha	10,000	75,100
Advanced	Decaptcha	94,335	708,456

Table 4: Estimated number of registrations per day for various threat models, using the best Sphinx and Decaptcha models, and assuming a limit of 10 audio captcha downloads per IP per day.

Abusing redundancy. While the Decaptcha model uses multiple samples of each captcha during training, neither it nor the Sphinx model takes advantage of this redundancy to improve their performance. Intuitively, however, this redundancy should be useful: if the model is unable to guess a captcha, it may still be able to guess it with a second or third sample. Thus, we consider two approaches to exploiting this redundancy at prediction time. In the first case, we look at N samples of the captcha, guess sequences from these, and then select the most frequently guessed sequence as our response. In the second case, we again guess sequences for N samples, but this time look at the digits individually, selecting the most frequent digit found at each position.

Table 3 shows the results of allowing models to use additional samples of each captcha. The numbers for each N were produced by running the models on 1000 random samples of size N from the test data, and averaging accuracy across these trials. For all models, increasing the number of audio samples increases the model accuracy, and performance seems to be similar whether selecting the most frequent digits (the -d models) or the most frequent sequences. Because of *Decaptcha*'s high accuracy, it can achieve a 13 point boost in accuracy (from 75.1% to 88.2%) by simply looking at two samples per captcha instead of one.

Registrations per day. Table 4 shows the estimated number of registrations per day various attackers could achieve, making the conservative assumption that a limit of 10 audio captchas per IP per day was imposed. Under this scenario, a low threat attacker using off-the-shelf software and Tor to supply IP addresses could likely get only 150 registrations per day. Increasing the IP pool could yield about 1,000 registrations per day for a Medium threat attacker, while building a tool like Decaptcha could yield tens or hundreds of thousands of registrations per day. These results suggest that a low knowledge attacker with only a small IP pool is probably not

		Samples				
Err	DL	1	2	3	4	5
1	10	**5.7**	3.7	2.7	1.9	2.0
1	20	**8.6**	6.3	5.0	4.6	3.9
1	50	**11.5**	9.1	10.8	10.4	9.5
1	100	12.0	10.0	15.9	**18.9**	18.0
1	200	12.0	10.1	18.9	29.9	**32.7**
2	10	**7.2**	4.3	2.8	2.0	2.0
2	20	**13.0**	8.4	5.6	4.9	4.0
2	50	**21.4**	15.6	14.2	11.6	9.9
2	100	**23.8**	19.6	25.2	23.6	19.7
2	200	**23.9**	20.2	35.5	43.2	38.8

Table 5: Average number of captchas broken by Decaptcha by distinct IP when an IP address is refused after Err errors, and after DL downloads.

a substantial threat for eBay. However, large numbers of fake registrations could be produced by low knowledge attackers with larger IP pools, or attackers capable of developing a Decaptcha-like tool.

5 Mitigation

As mentioned previously, the most straightforward strategy for eBay to reduce the number of captchas that can be solved by an attacker would be to limit both the number of captcha downloads allowed to each IP address (download limit) and the number of times an IP address is allowed to submit an incorrect response (error limit).

To evaluate how this mitigation would improve the situation, we analyzed if it would be beneficial for the attacker to download additional samples of the same captcha, rather than always trying to guess the captcha from a single sample. We ran 100,000 simulations where the model classified as many captchas as it could before hitting either the download limit or the error limit. Table 5 shows the average number of captchas broken for each set of parameters.

In almost all cases, we found that it is more efficient to use only a single sample of each captcha. However, when the IP is banned after the first captcha error, and when a relatively large number of downloads per IP are allowed (e.g. 100), more captchas can be solved by exploiting extra samples. This corresponds to our intuition that when there is a high cost for failing on a captcha, the added accuracy from the additional samples will be useful. Still, under most conditions, using a single sample is the optimal strategy, so allowing the download of multiple samples of the same captcha is only an issue because it allows the attacker to easily construct a training corpus.

We can't determine the exact threshold eBay should use to thwart attackers while not preventing real users from registering because we don't have access to the necessary eBay statistics. However, this table suggests that the best mitigation technique is to reduce the number of attempts that a user is allowed before the system is locked. A less efficient but less drastic mitigation is to limit the number of captchas by IP. Enforcing a limit of 10 attempts, which seems reasonable, will prevent the attacker from registering more than 10 times with the same IP. Of course implementing these mitigations is not a substitute for a good captcha system but it helps to restrain the attacker. In particular these restrictions will force the attacker to use multiple IPs which will increase the attack entry cost.

Another mitigation technique would be to use variable length captchas: this would prevent the attacker from discarding the captchas that do not have the correct number of digits, and would make tuning the segmentation more difficult.

6 Additional Related Work

The first discussion of the captcha idea appears in [12], though the term CAPTCHA was coined in [16]. Text/image based captchas have been studied extensively [9, 10, 2] and there is a long record of successful attempts at breaking captchas of popular sites [3]. For example in March 2008, a method to break 60% of MSN visual captchas was disclosed [18]. One of the most famous visual captcha breakers is PWNtcha [7].

7 Conclusion

Our tool Decaptcha can break 75% of eBay audio captchas by analyzing energy peaks. Enforcing limits on downloads or incorrect answers can slow the attack, but truly countering Decaptcha will require more difficult captchas. However, increasing the difficulty of audio captchas for machines may also increase the difficulty for humans. Thus, we are planning an study to evaluate the difficulty of different types of audio captchas for both humans and machines, with a particular eye towards how audio captchas are solved by non-native speakers.

References

[1] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us: Automated identity theft attacks on social networks. In *18th International World Wide Web Conference (WWW 2009)*, 2009. 1

[2] K Chellapilla and P Simard. Using machine learning to break visual human interaction proofs. In MIT Press, editor, *Neural Information Processing Systems (NIPS)*, 2004. 6

[3] Dancho Danchev. Microsoft's captcha successfully broken. blog post http://blogs.zdnet.com/security/?p=1232, May 2008. 6

[4] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *In Proceedings of the 13th USENIX Security Symposium*, pages 303–320, 2004. 3

[5] Anne Eisenberg. New puzzles that tell humans from machines. http://www.nytimes.com/2009/05/24/business/24novelties.html?_r=1&ref=technology, May 2009. 1

[6] Julian B. Grizzard, Vikram Sharma, Chris Nunnery, Brent ByungHoon Kang, and David Dagon. Peer-to-peer botnets: Overview and case study. In *USENIX Workshop on Hot Topics in Understanding Botnets (HotBots)*, 2007. 3

[7] Sam Hocevar. Pwntcha captcha decoder. web site, http://sam.zoy.org/pwntcha. 6

[8] jochem. devoicecaptcha. web site http://vorm.net/captchas, 2006. 4

[9] P Simard K Chellapilla, K Larson and M Czerwinski. Building segmentation based human-friendly human interaction proofs. In Springer-Verlag, editor, *2nd Int'l Workshop on Human Interaction Proofs*, 2005. 2, 6

[10] P Simard K Chellapilla, K Larson and M Czerwinski. Designing human friendly human interaction proofs. In ACM, editor, *CHI05*, 2005. 6

[11] C. Kreibich, C. Kanich, K. Levchenko, B. Enright, G. Voelker, V. Paxson, and S. Savage. Spamcraft: An inside look at spam campaign orchestration. In USENIX, editor, *LEET*, 2009. 3

[12] Moni Naor. Verification of a human in the loop or identification via the turing test. Available electronically: http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps, 1997. 1, 6

[13] Phillip Porras, Hassen Saidi, and Vinod Yegneswaran. An analysis of conficker's logic and rendezvous points. Technical report, SRI, 2009. 3

[14] Rubén Santamarta. Breaking gmail's audio captcha. Web site : http://blog.wintercore.com/, March 2008. 1, 2

[15] Jennifer Tam, Jiri Simsa, Sean Hyde, and Luis von Ahn. Breaking audio captchas. In *Advances in Neural Information Processing Systems*, 2008. 1, 4

[16] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In Sringer, editor, *Eurocrypt*, 2003. 6

[17] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. Sphinx-4: A flexible open source framework for speech recognition. *Sun Microsystems Technical Report*, (TR-2004-139), November 2004. 2, 4

[18] Jeff Yan and Ahmad Salah El Ahmad. A low-cost attack on a microsoft captcha. Ex confidential draft http://homepages.cs.ncl.ac.uk/jeff.yan/msn_draft.pdf, 2008. 2, 6

[19] Li Zhuang, John Dunagan, Daniel R. Simon, Helen J. Wang, Ivan Osipkov, Geoff Hulten, and J.D. Tygar. Characterizing botnets from email spam records. In Usenix, editor, *LEET*, 2008. 3