CAPTCHAs: Humans vs. Bots

Aleksey Kolupaev and Juriy Ogijenko
OCR Research Team

A Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) offers a way for Web service providers to make some conclusions about whether a "user" is human or robot. In fact, you see one of its incarnations almost every day on the Web: the most popular way of protecting Internet forms is to generate a special picture made up of letters and numbers and then ask the user to rekey it in a special box, the idea being that it's extremely hard to teach a bot how to recognize letters written in graphics.

Other types of CAPTCHAs related to the human mind that are similarly hard for computers to reproduce (such as abstract thinking, object detection and classification, and so forth) can also be used for CAPTCHAs, but in this article, we focus on images.

Optical character recognition

The optical character recognition (OCR) class of systems is specifically designed to recognize what's written in a picture. OCR software uses a lot of tested practices and approaches to recognize the text written in an image, and hackers also use OCR tools and technologies to crack CAPTCHAs.

For our work in OCR research, we use the term *recognizer* as part of a CAPTCHA hacking tool that returns what's written in a picture with a probability that we call *accuracy*, which measures the recognizer's success. If a CAPTCHA consists of four numbers in a picture, for example, the simplest recognizer—using random numbers—has 1 in 10,000 chances to match it, or 0.01 percent accuracy. But if a talented OCR developer writes a genius piece of software, the recognizer could theoretically have 100 percent accuracy.

The process of CAPTCHA recognition is a combination of efforts, approaches, and software that attempts to increase accuracy to an acceptable level. Of course, it's hard to define this level, but we've found that it often starts from 5 percent accuracy.

The real world

Web services not only check CAPTCHA code but can also make some heuristic conclusions from it, such as banning "request IP address" if the user enters the CAPTCHA code incorrectly more then *x* times. In fact, most Web services implement this technique, so recognition (even with 100 percent accuracy, which is almost unheard of) doesn't guarantee success for a bot attack. At the very least, an attacker just needs a network of anonymous proxies, which is quite easy to acquire but increases the endeavor's cost.

A CAPTCHA's effectiveness works only to a point: it's possible for a hacker to hire a group of people, buy an anonymous proxy list, and create just as many accounts as needed to "break" it. This solution costs several cents per picture and is even sold as a service in some parts of the hacker community. After all, a CAPTCHA's purpose is to make the price of hacking the form submission process (or whatever is protected against bots) as high as possible, but at the end of the day, any CAPTCHA is vulnerable to hackers, regardless of the image-generating algorithm. However, if a hacker needs to hire a human "recognizer" to do that, we can assume the CAPTCHA-generating tool is perfect.

An automated hack system has several advantages: it works for an infinite length of time for free, is scalable, and often (but not always) recognizes images faster. However, its accuracy is lower so the system must send many more queries to the server to get a correct response. But is there any way to make OCR software a nightmare for the bad guys? The answer is yes, but to do that, we have to know how to break CAPTCHAs in the first place.

Breaking a CAPTCHA

The first thing to keep in mind is that breaking a CAPTCHA has two main parts: segmentation and character recognition. Think of it from a divide-and-conquer approach: if what's written in a CAPTCHA image consists of parts, then the recognizer will try to see those parts in the same way (as different pieces). A *seg-*

mentor thus chunks the image into letters and then passes each chunk to a character recognizer, which in turn attempts to learn what character is written in the given picture.

A study from Kumar Chellapilla's group (http://research.microsoft.com/~kumarc/) at Microsoft Research focused on CAPTCHA letters with distortion and noise and found that a neural network could recognize a single character much easier than a human could. This fact didn't depend on font style, rotation, or distortion, which means if an attacker knows how to build a neural network, the only thing we can do as a defense mechanism is make the characters hard to separate.

Let's review some popular methods of making CAPTCHAs difficult to break.

Font tricks

Font tricks include using artistic fonts, different font styles, upper and lowercase versions together, rotation, and distortion. This approach is popular because it's fairly easy to do, but it affects a neural network's complexity and the time spent training it. In most cases, font tricks also decrease the image's readability—remember, the legitimate human user still needs to be able to rekey it. Too fanciful fonts with distortion and noise make the picture almost illegible.

On the other hand, if a CAPTCHA uses a simple font without distortion, hackers can just use mask correlation (an approach based on a bitwise comparison of the image with some predefined template) without having to do anything with the neural network.

Noise

Adding noise to a CAPTCHA is another popular approach: it often resembles salt and pepper (if it uses lots of black and white), but it can have colors, dots, lines, circles, and rectangles, which can be hard to break. This protection method adds one more step before segmenting because to remove noise hackers must write a *violator* to find a specific property that distinguishes letters—for instance, color, line width, border style, position, or size. The key here is to make the noise resemble letters as much as possible so that hackers don't have an easy way to define and use this property. However, just as with font tricks, using a large amount of noise can lead to harder readability, making the CAPTCHA strong but useless for legitimate users.

Color model

Using a color model is a bit tricky because a bad one actually helps hackers separate letters. The main rule is the same as for noise: objects of different colors must be as similar as possible, otherwise the different colors will easily differentiate anything legible from the background. Keep in mind, though, that this feature can severely affect colorblind users; it isn't the only problem users with disabilities can experience when using CAPTCHAs, as the W3C covers (www.w3.org/TR/turingtest/).

Overlap

Letter overlap is the most powerful weapon available in CAPTCHA technology: if letters overlap and are still readable, this technique represents a very serious obstacle for recognizers. However, if the CAPTCHA uses different colored letters, the overlapping might not make any sense because it'll be very easy to distinguish letters based on color. As with any other obfuscation approach, this one is also a compromise between user-side readability and effectiveness against OCR systems.

Letters in general

So how many and which letters should a CAPTCHA use? Letters can look similar, especially after we apply some distortion and noise—for example, look at C vs. G or O vs. 0. Depending on the font used, it can be extremely difficult for a human or a bot to spot the difference. Another question to consider is whether the CAPTCHA should use meaningful words or just a set of letters. Even if some letters aren't perfectly clear, the human can still guess the whole word, but so can a bot—it isn't hard for hackers to use a dictionary tool to estimate the probability of alternatives. What about the number of letters? Let's say we have five: a bot can recognize them with a 0.85 percent probability, which means the probability of recognizing the entire picture will be $0.85^5 = 0.44$; for eight letters, accuracy shifts to $0.85^8 = 0.27$. Clearly, the more letters used, the less accurate the bot will be in its attempts, but this also adds a layer of difficulty for legitimate users as well. Ultimately, it's harder for hackers to split an image if they don't know the total number of letters in the picture, so a fixed number isn't as good as varying numbers.

Examples

Figures 1a and 1b show examples of good and bad CAPTCHA images, respectively. Most good examples use letter overlapping and/or noise, which make it harder for hackers or bots to separate the letters. The bad examples simply aren't as thorough: Fotolog, for example, has some letters overlapping, but this seems more accidental than anything else. Paypal is quite primitive, ditto Friendster and digg.com. We were surprised with Microsoft's live.com CAPTCHA because the generator is quite powerful and has serious amounts of noise tying the letters together, but it seems as if the noise level is low (probably because live.com is new, and Microsoft is

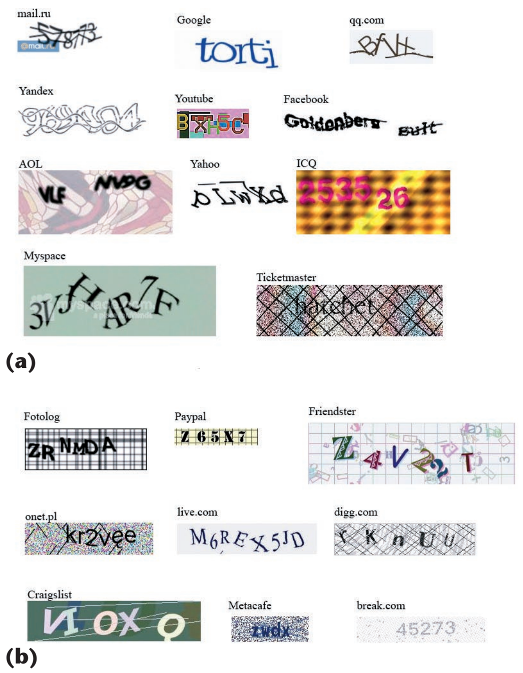

Figure 1. CAPTCHAs. The (a) good and (b) bad image examples differ in their use of obfuscating techniques such as letter overlap and noise.

expecting a lot of new registrations). Craigslist uses overlapping letters, but they're hard to read, so the site uses different colors in the fonts.

We've covered the most popular CAPTCHA approaches here but not all of them. Most users know how to use the more commonly used graphical CAPTCHAs, so designers must have serious reasons to migrate to other, more difficult-to-break, approaches. After all, CAPTCHAs aim to determine legitimate users, not alienate them.

The world's top IT science centers—including AT&T (http://yann.lecun.com/exdb/lenet/), Carnegie Mellon University (www.captcha.net), the University of California, Berkeley (www.eecs.berkeley.edu/Research/Projects/CS/vision/shape/), Simon Fraser University (www.cs.sfu.ca/~mori/research/gimpy/, and Microsoft Research (http://research.microsoft.com/asirra/)—are working on the challenges related to creating unbreakable and usable CAPTCHAs. Although the field has come a long way, there's still a lot of interesting research to be done. □

***Aleksey Kolupaev's** technical interests include Web projects and their interfaces and architecture. He has an MS in system analysis and project management from the National Technical University "KhPI" in Kharkiv, Ukraine. Contact him at kolupaev@gmail.com.*

***Juriy Ogijenko's** technical interests include AI-based software development, image processing, programming, and Web projects. He has an MS in mathematical research from the National Technical University "KhPI" in Kharkiv, Ukraine. Contact him at juriy.ogijenko@gmail.com.*