Machine Learning Attacks
Against the Asirra CAPTCHA

Philippe Golle
Palo Alto Research Center
Palo Alto, CA 94304, USA
pgolle@parc.com

ABSTRACT

The Asirra CAPTCHA [7], proposed at ACM CCS 2007, relies on the problem of distinguishing images of cats and dogs (a task that humans are very good at). The security of Asirra is based on the presumed difficulty of classifying these images automatically.

In this paper, we describe a classifier which is 82.7% accurate in telling apart the images of cats and dogs used in Asirra. This classifier is a combination of support-vector machine classifiers trained on color and texture features extracted from images. Our classifier allows us to solve a 12-image Asirra challenge automatically with probability 10.3%. This probability of success is significantly higher than the estimate of 0.2% given in [7] for machine vision attacks. Our results suggest caution against deploying Asirra without safeguards.

We also investigate the impact of our attacks on the *partial credit* and *token bucket* algorithms proposed in [7]. The partial credit algorithm weakens Asirra considerably and we recommend against its use. The token bucket algorithm helps mitigate the impact of our attacks and allows Asirra to be deployed in a way that maintains an appealing balance between usability and security. One contribution of our work is to inform the choice of safeguard parameters in Asirra deployments.

Categories and Subject Descriptors

K.6.5 [**Computing Milieux**]: Management of Computing and Information Systems—*Security and Protection*

General Terms

Security

Keywords

CAPTCHA, reverse Turing test, machine learning, support vector machine, classifier.

1. INTRODUCTION

The Asirra CAPTCHA [7], proposed at ACM CCS 2007, relies on the problem of distinguishing images of cats and dogs (Asirra stands for "Animal Species Image Recognition for Restricting Access"). An Asirra challenge consists of 12 images, each of which is of either a cat or a dog. To solve the CAPTCHA, the user must select all the cat images, and none of the dog images. This is a task that humans are very good at. According to [7], Asirra "can be solved by humans 99.6% of the time in under 30 seconds". The usability of Asirra is a significant advantage compared to CAPTCHAs [9] based on recognizing distorted strings of letters and numbers.

The security of Asirra is based on the presumed difficulty of classifying images of cats and dogs automatically. As reported in [7], evidence from the 2006 PASCAL Visual Object Classes Challenge suggests that cats and dogs are particularly difficult to tell apart algorithmically. A classifier based on color features, described in [7], is only 56.9% accurate. The authors of [7] conjecture that "based on a survey of machine vision literature and vision experts at Microsoft Research, we believe classification accuracy of better than 60% will be difficult without a significant advance in the state of the art". With a 60% accurate classifier, the probability of solving a 12-image Asirra challenge is only about 0.2%.

In this paper, we describe a classifier which is 82.7% accurate in telling apart the images of cats and dogs used in Asirra. This classifier allows us to solve a 12-image Asirra challenge with probability 10.3%. This success probability is significantly higher than the 0.2% estimate given in [7] for machine vision attacks. While our success rate may appear low in absolute terms, it nevertheless poses a serious threat to Asirra if additional safeguards are not deployed to prevent machine adversaries from requesting, and attempting to solve (at virtually no cost), too many CAPTCHAs.

Our classifier is a combination of two support-vector machine [5] (SVM) classifiers trained on color and texture features of images. The classifier is entirely automatic, and requires no manual input other than the one-time labelling of training images. Using 15,760 color features, and 5,000 texture features per image, our classifier is 82.7% accurate. The classifier was trained on a commodity PC, using 13,000 labelled images of cats and dogs downloaded from the Asirra website [1].

We also investigate the impact of our attacks on the *partial credit* and *token bucket* algorithms proposed in [7]. The partial credit algorithm weakens Asirra considerably and we recommend against its use. The token bucket algorithm

helps mitigate the impact of our attacks and allows Asirra to be deployed in a way that maintains an appealing balance between usability and security. One contribution of our work is to inform the choice of safeguard parameters in Asirra deployments.

Beyond this immediate contribution, we also hope that our paper will contribute to the popularization of machine learning techniques, for both offensive and defensive purposes, in the security community. Machine learning and other artificial intelligence techniques have not so far been widely used in cryptographic attacks. Yet recent work suggests that these techniques are powerful tools for the cryptanalyst's arsenal. SAT solvers, for example, have been used to find collisions in hash functions [13] and defeat authentication schemes [8]. Object recognition algorithms [14] were used in very successful breaks of the text-based Gimpy and EZ-Gimpy CAPTCHAs. The machine learning classifiers of the type used in this paper, likewise, will hopefully find broader applications in computer security.

Organization. We describe our SVM classifiers in section 2, using color features (section 2.1), texture features (section 2.2) and both in combination (section 2.3). We discuss the use of these classifiers in attacking Asirra in section 3. Section 3.1 investigates the impact of our attacks on the *partial credit* and *token bucket* algorithms of [7]. We briefly discuss other counter-measures that may help mitigate our attack in section 3.2. Finally, we review related work in section 4 and conclude in section 5.

2. SUPPORT VECTOR MACHINE CLASSIFIERS FOR ASIRRA IMAGES

Asirra relies on a large and growing database of some 3,000,000 images of cats and dogs licensed from the adoption service Petfinder.com. The images displayed by Asirra are 250-by-250 pixels. In the majority of images, there is either a single cat or a single dog. Some images contain multiple cats or multiple dogs. In a very few images, there is no recognizable animal, or else there is both a cat and a dog (these images cannot be classified according to the rules of Asirra).

Image collection. We collected 13,000 distinct images from the Asirra implementation publicly available on the Asirra website [1]. The website serves Asirra CAPTCHAs that consist of 12 images selected at random (according to [6]) from the entire Asirra image database. We wrote a script to automatically refresh the website and download the 12 images in the new Asirra CAPTCHA obtained after each refresh. Over the course of a night, our script refreshed the website approximately 1,100 times and downloaded just over 13,000 images. To avoid duplicates, every image was saved in a file named after a hash of its pixels (we detected and discarded 6 duplicate images). Other than duplicates, no images were deleted, filtered or otherwise selected.

The collection of 13,000 images thus obtained is a representative, unbiased sample of Asirra images, since "the Asirra service selects images randomly from [Asirra's] entire image database for each challenge" [6]. The Asirra authors conjecture that "Photos have a wide variety of backgrounds, angles, poses, lighting, and so forth – factors that make accurate automatic classification difficult" [7]. We have every reason to believe that our subset of 13,000 images offers a similar diversity of factors.

Manual classification. The next step was to manually classify the 13,000 images in our collection into 3 classes: Cat, Dog and Other. The Cat and Dog classes are self-explanatory. The Other class was for images which either contained no recognizable animal, or contained both a cat and a dog. Manual classification was followed by a manual verification step, in which 159 misclassified images (1.2% of the total) were detected and moved to the correct category. After verification, we obtained 6,403 images of cats (49.3%), 6,466 images of dogs (49.7%) and 131 other images (1.0% of the total). In the rest of our work, we kept only the images of cats and dogs and discarded the other images.

Building a classifier. We experimented with different color and texture features computed from images. These features are described in the rest of this section. We trained a support vector machine (SVM) classifier [5] with each set of features. SVM classifiers were selected for their ability to extract linear combination of features, their predictive power, and their computational scalability. We refer the reader to [10] for an excellent introduction to SVM (chapter 12), and a comparison of the characteristics of SVMs and other learning methods (page 313). In short, a SVM is a supervised learning method which constructs an optimal linear boundary (or separating hyperplane) between two classes. This hyperplane is optimal in the sense that it maximizes the distance, or margins, between the hyperplane and the two classes on each side of it (an error penalty accounts for misclassified points, when the two classes are not perfectly linearly separable). The power of SVM classifiers comes from the fact that the linear boundary is not computed directly in feature space, but in a transformed, higher-dimensional version of the feature space. The transformation is represented, loosely speaking, by a kernel function. Linear boundaries in the transformed space produce non-linear boundaries when mapped back to the original feature space.

Measuring accuracy. We measured the accuracy of our SVM classifiers using 5-fold cross-validation on random subsets of our image collection. Cross-validation operates by dividing a subset of images into 5 randomly chosen partitions; 4 of these partitions are used for training while the remaining one is used for validation. We report results using subsets of various sizes (5,000 and 10,000 images), to show the influence of the size of the training sample on the accuracy of our classifier. The accuracy reported for our classifiers in the following sections is the average accuracy (and its standard deviation) over the 5 experiments of 5-fold cross-validation. We note that all our subsets of images, and all the partitions used for cross-validation were generated at random to avoid any bias that might affect our results.

SVM implementation. We trained our SVM with a radial basis kernel. This kernel defines the inner product of two feature vectors v and v' as

$$K(v, v') = \exp\left(-\gamma |v - v'|^2\right).$$

The parameter γ was tuned with 5-fold cross-validation to approximately achieve the best test error performance. We found that $\gamma = 10^{-3}$ worked well for color features and $\gamma = 10^{-1}$ worked well for texture features. We used the LIBSVM [3] Java implementation of SVM. We rewrote parts of

Color features					# Images		Classifier accuracy	
Feature set	N	C_h	C_s	C_v	# features	Total	Training set	mean	stdev
F_1	1	10	10	10	1,000	5,000	4,000	**67.3 %**	1.6
F_2	3	10	8	8	5,760	5,000	4,000	**74.6 %**	1.1
F_3	5	10	6	6	9,000	5,000	4,000	**74.6 %**	0.6
F_3	5	10	6	6	9,000	10,000	8,000	**75.7 %**	0.7

Table 1: **Accuracy of SVM classifiers trained on color features extracted from Asirra images. The color features are described in section 2.1. The accuracy of the classifier is the fraction of cat and dog images classified correctly in the test set.**

Color features		# Images		Classifier accuracy	
Feature set	# features	Total	Training set	mean	stdev
$F_1 \cup F_2 \cup F_3$	15,760	5,000	4,000	**76.3 %**	0.9
$F_1 \cup F_2 \cup F_3$	15,760	10,000	8,000	**77.1 %**	0.6

Table 2: **Accuracy of SVM classifiers trained on a combination of color features.**

the LIBSVM library to make more economical use of memory for vectors of boolean features. All computations were done on a commodity desktop PC running Windows XP with dual 3.40 GHZ CPUs and 3.00 GB of RAM.

2.1 Color Features

Recall that an Asirra image is 250-by-250 pixels. We divide the image into N vertical and N horizontal strips of equal width. We thus obtain a division of the image into a grid of N^2 cells. Each cell is a square of width and height $250/N$ (rounded to the nearest integer).

We also partition the color space. We use the HSV (hue, saturation, value) model of color, since it is closer to human perception of color, and thus easier to interpret, than the RGB (red, green, blue) model. We subdivide the hue channel of the color spectrum into C_h bands of equal width, the saturation channel into C_s bands of equal width and the value channel into C_v bands of equal width. Altogether, this gives us a partition of the color space into $C_h C_s C_v$ color regions.

Informally, the feature vector associated with an image indicates, for every cell in the image and every color region, whether there is at least one pixel in the cell which belongs to the color region. Note that these features are boolean. Unlike color histograms, they do not indicate *how many* pixels in a given cell fall within a certain color region, but only whether *one or more* pixel in the cell falls in the color region. Our experiments show that these boolean features yield more accurate classifiers than color histograms, in addition to being more efficient.

More precisely, the feature vector $\mathcal{F}(N, C_h, C_s, C_v)$ is a boolean vector of length $N^2 C_h C_s C_v$. The boolean feature associated with cell $(x, y) \in [1, \ldots, N] \times [1, \ldots, N]$ and color region $(h, s, v) \in [1, \ldots, C_h] \times [1, \ldots, C_s] \times [1, \ldots, C_v]$ takes the value 1 (or true) if there is one or more pixel in cell (x, y) of a color that belongs to color region (h, s, v). Otherwise, the feature takes the value 0 (false).

We trained SVM classifiers with these color features, and measured their accuracy using 5-fold cross-validation, as explained above. Our results are given in Table 1 for various values of the parameters N, C_h, C_s, C_v and for training sets of various sizes. For example, the feature set $F_3 = \mathcal{F}(5, 10, 6, 6)$ consists of 9,000 color features obtained with

a division of images into 25 cells ($N = 5$) and a division of the color space into 360 color regions ($C_h = 10$, $C_s = 6$ and $C_v = 6$). With 4,000 training images, an SVM classifier using the feature set F_3 is on average 74.6% accurate (over the 5 experiments of 5-fold cross-validation). With 8,000 training images, the accuracy of this classifier increases to 75.7 %.

Combining color features computed on cells of various sizes further improves the accuracy of our classifier. We experimented with the union of the three feature sets $F_1 = \mathcal{F}(1, 10, 10, 10)$, $F_2 = \mathcal{F}(3, 10, 8, 8)$ and $F_3 = \mathcal{F}(5, 10, 6, 6)$ of Table 1. The total number of color features in $F_1 \cup F_2 \cup F_3$ is 15,760. The accuracy of a classifier using these features is shown in Table 2. With 4,000 training images, the classifier is 76.3 % accurate. With 8,000 training images, it is 77.1 % accurate.

Color features and classifier accuracy. We observed in our experiments that SVM classifiers trained on boolean color features are more accurate than those trained on color histograms. This finding runs counter to intuition: boolean features, which record only the presence or absence in an image of pixels belonging to a certain region of the color spectrum, encode strictly less information than color histograms, which also record the number of such pixels. We advance two hypotheses for why boolean features outperform color histograms. The first is that boolean color features, unlike color histograms, are scale-independent: they record the presence or the absence of, say, the green of a cat's eye or the pink of a dog's tongue regardless of the size of the eye or tongue in the picture. Our second hypothesis is that the distribution of boolean features is much more regular (only two values are possible) than the distribution of real-valued color histograms (in which the range of possible values may cover several orders of magnitude). The regularity of boolean features facilitates the tuning of SVM parameters (notably γ), which results in superior accuracy.

Predictive power of individual color features. We observe that individual boolean color features, in isolation, fail to distinguish cats from dogs with any accuracy. The graph in Figure 1 shows the 1,000 boolean color features in $F_1 = \mathcal{F}(1, 10, 10, 10)$, plotted according to the fraction
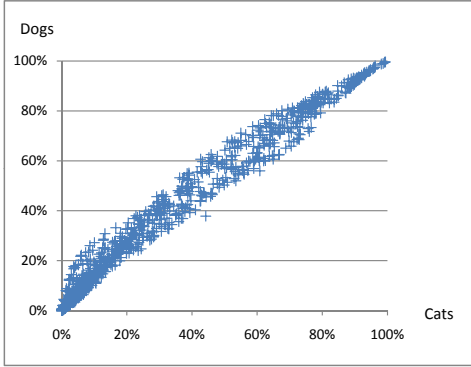
Figure 1: This graph shows the 1,000 boolean color features in $F_1 = \mathcal{F}(1, 10, 10, 10)$, plotted according to the fraction of cats (horizontal axis) and the fraction of dogs (vertical axis) for which the feature evaluates to "True". Note that features are clustered along the diagonal.

Figure 2: Example of texture tiles (5-by-5 pixels) extracted from images of cats and dogs. The classifier of section 2.2 relies on such texture tiles.

of cats (horizontal axis) and the fraction of dogs (vertical axis) for which the feature evaluates to "True". The features are clustered along the diagonal, which indicates that the ability of any given color feature to distinguish between cats and dogs is very weak. Nevertheless, an SVM classifier can harness the aggregate power of the color features to produce more accurate predictions. The success of our classifier does not come from the careful selection of a few colors with high predictive values, but rather from the combination of a large number of weakly predictive features.

2.2 Texture Features

Approaches to texture recognition can be broadly divided into two categories. The first category takes a *statistical* approach to texture computation. Texture is defined via quantitative measurements of intensity in different regions of the image (e.g. by convolution with a Gabor filter [11]). The second category takes a *structural* approach, and defines texture as a set of texture tiles (also called texels) in repeated patterns. We experimented with both approaches, and found that structural measurements of texture produced more accurate classifiers.

We start with an informal presentation of our structural approach to texture recognition. We extract small sub-images (5-by-5 pixels) from training images of cats and dogs. We call these sub-images *texture tiles*. Figure 2 shows examples of texture tiles extracted from Asirra images. We collect a set $\mathcal{T}$ of texture tiles of size $t = |\mathcal{T}|$, such that the distance between any two tiles in $\mathcal{T}$ is above a certain threshold (we define below our measure of distance between tiles). This ensures that the tiles in $\mathcal{T}$ are sufficiently diverse, and that there are no duplicate tiles. The feature vector associated with an image is the vector of distances between the image and each texture tile in $\mathcal{T}$ (we define below our measure of distance between an image and a tile). Finally, we train an SVM classifier with these feature vectors. More precisely, we proceed as follows.

Selection of texture tiles. We select random images of cats and dogs from the set of training images. We divide each image into vertical and horizontal strips of equal width (5 pixels). We thus obtain a division of each image into $(250/5)^2 = 2500$ feature tiles. Each feature tile is a square of 5-by-5 pixels. Let us denote $\mathcal{T}_0$ this initial set of candidate tiles. We define the distance between two tiles as the average Euclidian distance between the pixels of the tiles in RGB color space. From $\mathcal{T}_0$, we then compute a subset $\mathcal{T}$ of texture tiles iteratively as follows. Initially, $\mathcal{T}$ is empty. We consider in turn each tile $T \in \mathcal{T}_0$. If there already exists a tile in $\mathcal{T}$ whose distance to T is below a certain threshold δ, we discard T. Otherwise, we add the tile T to $\mathcal{T}$. We repeat this computation for all candidate tiles in $\mathcal{T}_0$ until we obtain a set $\mathcal{T}$ of size t. Note that the initial set $\mathcal{T}_0$ must be chosen of sufficiently large size to ensure the existence of a subset $\mathcal{T}$ of size t.

Feature vector. The feature vector associated with an image is the vector of distances between the image and each of the t texture tiles in $\mathcal{T}$. The distance between an image A and a texture tile $T \in \mathcal{T}$ is defined as follows. For $0 \leq i, j \leq (250 - 5)$, let us denote $A_{i,j}$ the square sub-image of A, of width 5 pixels and height 5 pixels, whose top left corner is the pixel of A in row i and column j. We define the distance $d(A_{i,j}, T)$ between a sub-image $A_{i,j}$ and a texture tile T as the maximum of the Euclidean distance between their pixels in RGB space. The distance between A and T is defined as $d(A, T) = \min_{i,j} d(A_{i,j}, T)$. Distances are normalized to the range $[0, 1]$.

Results. We trained an SVM classifier with these texture features, and measured its accuracy using 5-fold cross-validation. Our results are given in Table 3. The feature set G_1 consists of 1,000 features which record the distance of an image to 1,000 texture tiles. The texture tiles are selected such that the distance between any two of them is at least 40.0. With 4,000 training images, an SVM classifier using the feature set G_1 is 74.5 % accurate. We define a feature set G_2 which consists of 5,000 features which record the distance of an image to 5,000 texture tiles similarly selected.

Texture features			# Images		Classifier accuracy	
Feature set	# tiles	δ	Total	Training set	mean	stdev
G_1	1,000	40.0	5,000	4,000	**74.5 %**	2.0
G_2	5,000	40.0	5,000	4,000	**78.0 %**	1.9
G_2	5,000	40.0	10,000	8,000	**80.4 %**	0.9

Table 3: Accuracy of SVM classifiers trained on texture features extracted from Asirra images. The texture features are described in section 2.2. The accuracy of the classifier is the fraction of cat and dog images classified correctly in the test set.

Features	# Images		Classifier accuracy	
	Total	Training set	mean	stdev
$(F_1 \cup F_2 \cup F_3) + G_2$	5,000	4,000	**80.3 %**	1.4
$(F_1 \cup F_2 \cup F_3) + G_2$	10,000	8,000	**82.7 %**	0.5

Table 4: Accuracy of the combined outputs of the color classifier of section 2.1 and the texture classifier of section 2.2. The color classifier is given half the weight of the texture classifier (see section 2.3).
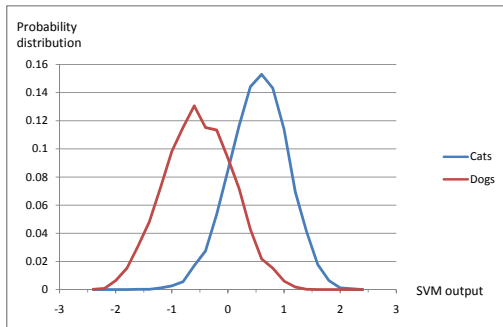

Figure 3: Probability distribution of the combined output of the color and texture classifiers of section 2.3.

Figure 4: The cats and dogs in our sample that are most cat-like and most dog-like, according to the classifier of section 2.3.

	Classified as cat	Classified as dog
Cats	4,271	729
Dogs	997	4,003

Table 5: Confusion matrix for the combined color and texture classifier of section 2.3.

Using this larger feature set and 4,000 training images, the accuracy of our SVM classifier increases to 78.0%. With 8,000 training images, it is 80.4 % accurate.

2.3 Combination of Color and Texture Features

The SVM classifiers of sections 2.1 and 2.2 produce for each image a real-valued estimate of whether the image is of a cat or of a dog. We mapped the "cat" class to the value 1.0 and the "dog" class to the value -1.0. Thus, an image which produces a positive output is labelled "cat" and an image which produces a negative output is labelled "dog". The output of different SVM classifiers can be combined simply by a weighted average of the estimates they produce. We combined in this way an SVM classifier which uses the set of color features $F_1 \cup F_2 \cup F_3$ (with weight $1/3$) and a second SVM classifier which uses the set of texture features G_2 (with weight $2/3$). The accuracy of this combination is given in Table 4. With a training set of 8,000 images, we obtain a classifier which is 82.7 % accurate. The confusion matrix of this classifier is in Table 5.

Figure 3 shows the probability distribution of the combined outputs of the color and texture SVM classifiers, for the "cat" and "dog" classes. Figure 4 shows the cats and dogs in our sample of 13,000 pets that are most cat-like and most dog-like, according to the combined classifier.

Accuracy versus completeness. We can achieve lower error rates if we allow the classifier to assign some images a "don't know" label. The quality of this 3-class classifier is measured by completeness (the fraction of images classified as either "cat" or "dog") and accuracy (the fraction of images in the "cat" and "dog" classes which are accurately classified). We turn our combined color and texture classifier into a 3-class classifier as follows. The classifier is parameterized by
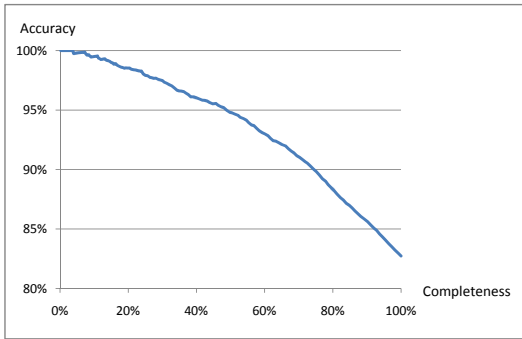
Figure 5: Accuracy versus completeness of the combined color and texture classifier of section 2.3.

a real-valued parameter $\epsilon \geq 0$. Images which produce an output smaller than $-\epsilon$ are labelled "dog". Images which produce an output between $-\epsilon$ and ϵ are labelled "don't know". Finally, images which produce an output larger than ϵ are labelled "cat".

Figure 5 shows a plot of the accuracy versus completeness of this classifier obtained for different values of the parameter ϵ. The base accuracy of the combined color and texture classifier, when classifying all images (completeness of $100\,\%$), is $82.7\,\%$. If the classifier can ignore half the images (completeness of $50\,\%$), its accuracy rises to $94.8\,\%$. For $20\,\%$ completeness, accuracy rises to $98.5\,\%$.

3. ATTACKING ASIRRA

In this section, we describe the application of the machine classifiers of section 2 to attacking Asirra. Recall that an Asirra challenge consists of 12 images of cats and dogs. To solve the challenge, one must identify correctly the subset of cat images.

Generic attack. A classifier with a success probability $0 < p < 1$ of correctly classifying a single Asirra image succeeds in solving a 12-image Asirra challenge with probability p^{12}. Our best classifier is $82.74\,\%$ accurate, which implies that we can solve an Asirra challenge completely automatically with probability $10.3\,\%$. Note that this success probability is independent of the probability distribution according to which Asirra challenges are generated.

Leveraging prior information. This success probability can be improved, if the attacker has knowledge of the prior probability distribution of Asirra challenges over the space $\{\mathrm{Cat}, \mathrm{Dog}\}^{12}$ (whatever that distribution may be). Let $C = \{I_1, \ldots, I_{12}\}$ denote a 12-image Asirra challenge and let $A = \{a_1, \ldots, a_{12}\}$, where $a_i \in \{\mathrm{Cat}, \mathrm{Dog}\}$ denote an assignment of the images $I_1, \ldots, I_{12}$ to the "cat" and "dog" classes. According to Bayes' rule,

$$\Pr[A|C] = \Pr[A]\Pr[C|A]/\Pr[C].$$

The attacker's goal is to compute $\max_A \Pr[A|C]$, or equivalently $\max_A (\Pr[A]\Pr[C|A])$. The first term, $\Pr[A]$ is the prior probability distribution that we are assuming is known to the attacker. The second term, $\Pr[C|A]$ can be estimated by the attacker as follows. Let us assume that the attacker uses a classifier $\mathtt{Cl}$ which produces real-valued outputs, as

in section 2.3. Let δ_{cat} denote the probability distribution of the output of $\mathtt{Cl}$ over images of cats, and δ_{dog} denote the probability distribution of the output of $\mathtt{Cl}$ over images of dogs. With this classifier, the attacker can estimate

$$\Pr[C|A] = \prod_{i\,|\,a_i=\mathrm{cat}} \delta_{\mathrm{cat}}(\mathtt{Cl}(I_i)) \prod_{i\,|\,a_i=\mathrm{dog}} \delta_{\mathrm{dog}}(\mathtt{Cl}(I_i)).$$

Example. The exact rules for creating Asirra challenges are not specified precisely in [7]. The basic rule, however, seems to be that the 12 images of an Asirra challenge are drawn uniformly at random, either from the full Asirra database of more than 3,000,000 images, or from a subset of images of pets located in the geographic vicinity of the user. If this assumption is correct, it implies that each image in an Asirra challenge is drawn independently at random from the "cat" class with probability q and from the "dog" class with probability $1 - q$. An attacker can learn the value of the parameter q by observing Asirra challenges. Our own measurements suggest $q \approx 0.5$ since we found approximately the same number of cats and dogs in our sample of the Asirra database. As explained above, we can leverage this information to compute the most likely assignment A for a given challenge: $\max_A(\Pr[A]\Pr[C|A])$. In this example, $\Pr[A] = q^w(1-q)^{12-w}$, where w is the number of cats in A. Using the combined color and texture classifier of section 2.3 to estimate $\Pr[C|A]$, we solve an Asirra challenge with probability $10.4\,\%$. This probability of success is only barely higher than that of the generic attack (10.3%). The reason is that, with the classifier of section 2.3, the generic attack already produces assignments that nearly follow a binomial distribution of cats and dogs.

Hypothetical example. Consider an hypothetical variant of Asirra in which every challenge contains exactly 6 images of cats and 6 images of dogs (this variant is *not* proposed in [7]). We now have $\Pr[A] = 0$ if the number of cats and dogs in A are not equal, and $\Pr[A] = 1/\binom{12}{6}$ otherwise. Using the Bayesian formula above, and the classifier of section 2.3, we can solve these variant Asirra challenges automatically with probability $23.8\,\%$. While this variant may be attractive from a usability point-of-view (users may solve Asirra challenges faster if they know they must find exactly 6 cats), our analysis shows that it is insecure and should be avoided.

3.1 Partial Credit Algorithm and Token Bucket Scheme

Two enhancements to Asirra are proposed in [7]. The first is a partial credit algorithm designed to improve the usability of Asirra for human users. The second is a token bucket scheme designed to harden Asirra against automated attacks. In this section, we study the impact of these enhancements on the classifier of section 2.3.

Partial credit algorithm (PCA). A user who correctly classifies 11 of the 12 images in an Asirra challenge is considered to have "nearly" solved the challenge. The user is placed in an intermediate state and presented with a second challenge. If the user solves or nearly solves the second challenge (i.e. identifies 11 or 12 images correctly), the user passes. Otherwise, the user fails and is returned to the default (non intermediate) state. Table 6 shows the impact of

# trials	Classifier success		Human success	
	no PCA	PCA	no PCA	PCA
1	10.3 %	10.3 %	83.4 %	83.4 %
2	19.5 %	26.2 %	97.2 %	99.6 %
3	27.8 %	38.0 %	99.5 %	99.9 %

Table 6: Impact of the partial credit algorithm on the success of the classifier of section 2.3. For comparison, the table also includes the human success rates reported in [7].

Enhancement	TB-refill	Classifier success rate
Token bucket	3	2.9 %
Token bucket	2	2.0 %
Token bucket	1	1.1 %
Token bucket + PCA	3	19.0 %
Token bucket + PCA	2	15.3 %
Token bucket + PCA	1	13.0 %

Table 7: Impact of the token bucket scheme on the success of the classifier of section 2.3. In [7], the parameter *TB-refill* is set to 3.

the partial credit algorithm on the success of the classifier of section 2.3 (for comparison, the table also includes the figures for the human success rate, taken from [7]). With PCA, the success rate of our automatic classifier is 38.0 % after 3 challenges. This is unacceptably high, and leads us to recommend that the partial credit algorithm should not be deployed with Asirra.

Token bucket scheme. A full description of the token bucket scheme can be found in [7]. In essence, the token bucket scheme punishes users who fail a lot of Asirra challenges. These users must solve correctly two Asirra challenges in close succession to be considered successful. The token bucket scheme is parameterized by a parameter *TB-refill*, which specifies how many chances the user is given to correctly solve a second CAPTCHA after solving the first one. A value *TB-refill* = 1 means that a user who has failed "too many" Asirra challenges must then solve two successive CAPTCHAs correctly to be considered successful. In [7], the value *TB-refill*=3 is suggested, which means the user is allowed 3 trials to solve a second CAPTCHA correctly. Table 7 shows the impact of the token bucket scheme on the success of the classifier of section 2.3. Our results suggest that PCA leads to weak security, even in combination with the token bucket scheme. On the other hand, Asirra appears reasonably secure with the parameter *TB-refill*=1, since our attack in that case is only 1.1% successful (of course, this parameter is bound to also significantly decrease the human success rate, and thus negatively impact the usability of Asirra).

3.2 Defenses

The best defenses against our machine learning attacks are IP monitoring schemes, which prevent an adversary from requesting, and attempting to solve, too many Asirra challenges. The token bucket scheme proposed in [7], and summarized above in section 3.1, is a clever instantiation of an

IP monitoring scheme. The strictest version of the token bucket scheme reduces the probability of solving an Asirra challenge automatically to 1.1% (with, however, a parallel reduction in the usability of Asirra for humans). The token bucket scheme could be further strengthened by requiring users to correctly solve more than two Asirra challenges in a row. Unfortunately, this would also negatively affect the ability of humans to pass Asirra challenges. Another approach to improving the security of Asirra is to increase the number of images used in challenges.

Distorting, warping or degrading the quality of the images is unlikely to do much to lower the accuracy of SVM classifiers based on color and texture features, since these features are largely unaffected by global image distortions. Using greyscale images, instead of color images, may decrease the accuracy of the color classifiers of section 2.1, but would likely have little effect on the texture classifiers of section 2.2. These techniques do not appear promising: they are unlikely to dent the effectiveness of automatic classifiers without also significantly reducing the usability advantage that is Asirra's greatest strength. They would amount to "the arms race found in text CAPTCHAs that [Asirra is] trying to avoid" [7].

4. RELATED WORK

A number of attacks have been reported against text-based CAPTCHAs. Mori and Malik [14] proposed object recognition algorithms that succeeded in recognizing words in the EZ-Gimpy CAPTCHA with probability 92% and in the Gimpy CAPTCHA with probability 33%. More recently, attacks have been reported in the popular press against the CAPTCHAs used by Yahoo! [15] and Google [16]. Unfortunately, few details are available and it is difficult to ascertain the validity of these attacks. Very recent work [17] gives a detailed description of character segmentation attacks against Microsoft and Yahoo! CAPTCHAs.

Beyond Asirra, there have been other proposals for user-friendly, clickable CAPTCHAs. For example, Lopresti [12] proposes asking users to select the right orientation of a page through a click. BotBarrier [2] asks users to click on a specified location in an image. The security of these proposals relies on new and relatively untested assumptions. It is not clear whether these assumptions will withstand the test of time.

Another approach to clickable CAPTCHAs was recently proposed by Chow et al. [4]. The approach consists of combining several textual CAPTCHAs into a grid of clickable CAPTCHAs (e.g. a 3-by-4 grid). The solution to the grid is the determination (e.g. by clicking) of the grid elements which satisfy some given requirement. For example, the user may be asked to identify in the grid the subset of CAPTCHAs which embed English words (assuming some, but not all, do). One advantage of this approach is that it relies on existing security assumptions about text-based CAPTCHAs that have been in use for a long time and have been the object of intense scrutiny.

5. CONCLUSION

We describe a classifier which is 82.7% accurate in telling apart the images of cats and dogs used in Asirra. This classifier allows us to solve a 12-image Asirra challenge with probability 10.3%. The weakness we have exposed in the

current implementation of Asirra cautions against deploying Asirra without additional safeguards. With appropriate safeguards, notably the token bucket scheme described in [7], we believe that Asirra continues to offer an appealing balance between security and usability. We hope that this work will contribute to the secure deployment of Asirra.

Acknowledgements

6. REFERENCES

[1] MSR Asirra: A Human Interactive Proof. On the Web at `http://research.microsoft.com/asirra/`

[2] BotBarrier.com. On the web at `http://www.botbarrier.com/`

[3] Chih-Chung Chang and Chih-Jen Lin. LIBSVM : a library for support vector machines, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

[4] R. Chow, P. Golle, M. Jakobsson, X. Wang and L. Wang. Making CAPTCHAs Clickable. In *Proc. of HotMobile 2008*.

[5] C. Cortes and V. Vapnik. Support-vector network. Machine Learning 20, 273–297, 1995.

[6] J. Douceur and J. Elson. Private communication.

[7] J. Elson, J. Douceur, J. Howell and J. Saul. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In *Proc. of ACM CCS 2007*, pp. 366–374.

[8] P. Golle and D. Wagner. Cryptanalysis of a Cognitive Authentication Scheme. In *Proc. of the 2007 IEEE Symposium on Security and Privacy*, pp.66–70. IEEE Computer Society

[9] Google CAPTCHA. On the web at `https://www.google.com/accounts/DisplayUnlockCaptcha`

[10] T. Hastie, R. Tibshirani and J. Friedman. The Elements of Statistical Learning (Data Mining, Inference, and Prediction). Springer Series in Statistics, 2001.

[11] P. Kruizinga, N. Petkov and S.E. Grigorescu. Comparison of texture features based on Gabor filters. In *Proc. of the 10th International Conference on Image Analysis and Processing (1999)*, pp. 142–147.

[12] D. Lopresti. Leveraging the CAPTCHA problem. In *Proc. of the Second International Workshop on Human Interactive Proofs*, pp. 97–110. Springer Verlag, 2005.

[13] I. Mironov and L. Zhang. Applications of SAT Solvers to Cryptanalysis of Hash Functions. In *Theory and Applications of Satisfiability Testing – SAT 2006*, pp. 102–115, 2006.

[14] G. Mori and J. Malik. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In *Proc. of the 2003 Conference on Computer Vision and Pattern Recognition*, pp. 134–144. IEEE Computer Society, 2003.

[15] SlashDot. Yahoo CAPTCHA Hacked (posted Jan 29, 2008). On the Web at `http://it.slashdot.org/it/08/01/30/0037254.shtml`

[16] Websense Blog (posted Feb 22, 2008). Google's CAPTCHA busted in recent spammer tactics. On the web at `http://securitylabs.websense.com/content/Blogs/2919.aspx`

[17] J. Yan and A. El Ahmad. A Low-cost Attack on a Microsoft CAPTCHA. To appear in *Proc. of ACM CCS 2008*.