Exploiting Machine Learning to Subvert Your Spam Filter

Blaine Nelson Marco Barreno Fuching Jack Chi Anthony D. Joseph
Benjamin I. P. Rubinstein Udam Saini Charles Sutton J. D. Tygar Kai Xia
University of California, Berkeley

Abstract

Using statistical machine learning for making security decisions introduces new vulnerabilities in large scale systems. This paper shows how an adversary can exploit statistical machine learning, as used in the SpamBayes spam filter, to render it useless—even if the adversary's access is limited to only 1% of the training messages. We further demonstrate a new class of *focused attacks* that successfully prevent victims from receiving specific email messages. Finally, we introduce two new types of defenses against these attacks.

1 Introduction

This paper demonstrates how attackers can exploit machine learning to subvert spam filters. Our attack strategies exhibit two key differences from previous work: traditional attacks modify spam emails to evade a spam filter, whereas our attacks interfere with the training process of the learning algorithm and *modify the filter itself*; and rather than focus only on placing spam emails in the victim's inbox, we subvert the spam filter to *remove legitimate emails* from the inbox. In this paper, we explore the implications of the *contamination assumption*: the adversary can control some of the user's training data.

An attacker may have one of two goals: expose the victim to an advertisement or prevent the victim from seeing a legitimate message. Potential revenue gain for a spammer drives the first goal, while the second goal is motivated, for example, by an organization competing for a contract that wants to prevent competing bids from reaching their intended recipient.

An attacker may have detailed knowledge of a specific email the victim is likely to receive in the future, or the attacker may know particular words or general information about the victim's word distribution. In many cases, the attacker may know nothing beyond which language the emails are likely to use.

When an attacker wants the victim to see spam emails, a broad *dictionary attack* can render the spam filter unusable, causing the victim to disable the filter (Section 3.2). With more information about the email distribution, the attacker can select a smaller dictionary of high-value features that are still effective. When an attacker wants to prevent a victim from seeing particular emails and has some information about those emails, the attacker can target them with a *focused attack* (Section 3.3).

We demonstrate the potency of these attacks and then present two defenses. The *Reject On Negative Impact (RONI) defense* tests the impact of each email on training and doesn't train on messages that have a large negative impact. The *dynamic threshold defense* dynamically sets the spam filter's classification thresholds based on the data rather than using SpamBayes' static choice of thresholds. We show that both defenses are effective in preventing some attacks from succeeding.

We focus on the learning algorithm underlying several spam filters, including SpamBayes (spambayes.sourceforge.net), BogoFilter (bogofilter.sourceforge.net), and the machine learning component of SpamAssassin (spamassassin.apache.org).[1] We target SpamBayes because it uses a pure machine learning method, it is familiar to the academic community [14], and it is popular with over 700,000 downloads. Although we specifically attack SpamBayes, the widespread use of its statistical learning algorithm suggests that other filters may also be vulnerable to similar attacks. However, some filters, such as SpamAssassin, use the learner only as one component of a broader filtering strategy.

Our experimental results confirm that this class of attacks presents a serious concern for statistical spam filters. A dictionary attack can make a spam filter unusable when controlling just 1% of the messages in the training set, and a well-informed focused attack can remove the target email from the victim's inbox 90% of the time. Of

[1]The primary difference between the learning elements of these three filters is in their tokenization methods.

our two defenses, one significantly mitigates the effect of the dictionary attack and the other provides insight into the strengths and limitations of threshold-based defenses.

2 Background

2.1 Training model

SpamBayes produces a classifier from labeled examples to label future emails. The labels are *spam* (bad, unsolicited email), *ham* (good, legitimate email), and *unsure* (SpamBayes isn't confident one way or the other). The classifier learns from a labeled *training set* of ham and spam emails.

Email clients use these labels in different ways—some clients filter email labeled as *spam* and *unsure* into "Spam-High" and "Spam-Low" folders, respectively, while other clients only filter email labeled as *spam* into a separate folder. Since the typical user reads most or all email in their inbox and rarely (if ever) looks at the spam/spam-high folder, the *unsure* labels can be problematic. If *unsure* messages are filtered into a separate folder, users may periodically read the messages in that folder to avoid missing important email. If instead *unsure* messages are not filtered, then the user faces those messages when checking the email in their inbox. Too much *unsure* email is almost as troublesome as too many false positives (ham labeled as *spam*) or false negatives (spam labeled as *ham*). In the extreme, if everything is labeled *unsure* then the user obtains no time savings at all from the filter.

In our scenarios, an organization uses SpamBayes to filter multiple users' incoming email[2] and trains on everyone's received email. SpamBayes may also be used as a personal email filter, in which case the presented attacks and defenses are likely to be equally effective.

To keep up with changing trends in the statistical characteristics of both legitimate and spam email, we assume that the organization retrains SpamBayes periodically (e.g., weekly). Our attacks are not limited to any particular retraining process; they only require that the attacker can introduce attack data into the training set somehow (the contamination assumption).

2.2 The contamination assumption

We assume that the attacker can send emails that the victim will use for training—the *contamination assumption*—but incorporate two significant restrictions: attackers may specify arbitrary email bodies but not headers, and attack emails are always trained as spam and not ham. We examine the implications of the contamination assumption in the remainder of this paper.

[2]We use the terms *user* and *victim* interchangeably for either organization or individual; the meaning will be clear from context.

How can an attacker contaminate the training set? Consider the following alternatives. If the victim periodically retrains on all email, any email the attacker sends will be used for training. If the victim manually labels a training set, the attack emails will be included as spam because they genuinely are spam. Even if the victim retrains only on mistakes made by the filter, the attacker may be able to design emails that both perform our attacks and are also misclassified by the victim's current filter. We do not address the possibility that a user might inspect training data to remove attack emails; our attacks could be adjusted to evade detection strategies such as email size or word distributions, but we avoid pursuing this arms race here.

Our focus on spam-labeled attack emails should be viewed as a restriction and not a necessary condition for the success of the attacks—using ham-labeled attack emails could enable more powerful attacks that place spam in a user's inbox.

2.3 SpamBayes learning method

SpamBayes classifies using token scores based on a simple model of spam status proposed by Robinson [14, 17], based on ideas by Graham [7] together with Fisher's method for combining independent significance tests [6].

SpamBayes tokenizes the header and body of each email before constructing token spam scores. Robinson's method assumes that the presence or absence of tokens in an email affect its spam status independently. For each token w, the raw token spam score

$$\mathrm{P}_S(w) \quad = \quad \frac{N_H N_S(w)}{N_H N_S(w) + N_S N_H(w)} \qquad (1)$$

is computed from the counts N_S, N_H, $N_S(w)$, and $N_H(w)$—the number of *spam* emails, *ham* emails, *spam* emails that include w and *ham* emails that include w.

Robinson smooths $\mathrm{P}_S(w)$ through a convex combination with a prior belief x, weighting the quantities by $N(w)$ (the number of training emails with w) and s (chosen for strength of prior), respectively:

$$f(w) \quad = \quad \frac{s}{s + N(w)}\, x + \frac{N(w)}{s + N(w)}\, \mathrm{P}_S(w) \ . \qquad (2)$$

For a new message E, Robinson uses Fisher's method to combine the spam scores of the most significant tokens[3] into a message score

$$I(E) = \frac{1 + H(E) - S(E)}{2} \in [0,1] \ , \quad (3)$$

$$H(E) = 1 - \chi^2_{2n}\left(-2 \sum_{w \in \delta(E)} \log f(w)\right) \ , \quad (4)$$

where $\chi^2_{2n}(\cdot)$ denotes the cumulative distribution function of the chi-square distribution with $2n$ degrees of freedom. $S(E)$ is defined like $H(E)$ but with $f(w)$ replaced by $1 - f(w)$. SpamBayes predicts by thresholding against two user-tunable thresholds θ_0 and θ_1, with defaults $\theta_0 = 0.15$ and $\theta_1 = 0.9$: SpamBayes predicts *ham*, *unsure*, or *spam* if I falls into the interval $[0, \theta_0]$, $(\theta_0, \theta_1]$, or $(\theta_1, 1]$, respectively.

The inclusion of an *unsure* category *spam* and *ham* prevents us from purely using misclassification rates (false positives and false negatives) for evaluation. We must also consider *spam-as-unsure* and *ham-as-unsure* emails. Because of the considerations in Section 2.1, *unsure* misclassifications of ham emails are nearly as bad for the user as false positives.

3 Attacks

3.1 Attack framework

In a previous paper, we categorize attacks against machine learning systems along three axes [1]. For concreteness, we will describe the taxonomy in the context of spam filtering, but it applies readily to other security applications. The axes of the taxonomy are:

Influence:	Causative or Exploratory
Security violation:	Integrity or Availability
Specificity:	Targeted or Indiscriminate

The first axis of the taxonomy describes the capability of the attacker: whether (a) the attacker has the ability to influence the training data that is used to construct the classifier (a *Causative* attack) or (b) the attacker does not influence the learned classifier, but can send new emails to the classifier, and observe its decisions on these carefully crafted emails (an *Exploratory* attack).

The second axis indicates the type of security violation caused: (a) to create false negatives, in which spam messages slip through the filter (an *Integrity* violation); or (b) to create false positives, in which ham messages are incorrectly filtered (an *Availability* violation).

The third axis refers to how specific the attacker's intention is: whether (a) the attack is *Targeted* to degrade the classifier's performance on one particular type of email or (b) the attack aims to cause the classifier to fail in an *Indiscriminate* fashion on a broad class of email.

Our focus is on Causative Availability attacks, which manipulate the filter's training data to increase false positives. We consider both Indiscriminate and Targeted attacks. In Indiscriminate attacks, enough false positives force the victim to disable the spam filter, or at least frequently search through spam/unsure folders to find legitimate messages that were filtered away. In either case, the victim is forced to view more spam. In Targeted attacks, the attacker does not disable the filter but surreptitiously prevents the victim from receiving certain types of email. For example, a company may wish to prevent its competitors from receiving email about a bidding process in which they are all competing.

3.2 Dictionary attacks

Our first attack is an Indiscriminate attack. The idea is to send *attack emails* that contain many words likely to occur in legitimate email. When the victim trains SpamBayes with these attack emails marked as spam, the words in the attack emails will have higher spam score. Future legitimate email is more likely to be marked as spam if it contains words from the attack email.

When the attacker lacks knowledge about the victim's email, one simple attack is to include an entire dictionary of the English language. This technique is the basic *dictionary attack*. We use the GNU `aspell` English dictionary version 6.0-0, containing 98,568 words.

A further refinement uses a word source with distribution closer to the victim's email distribution. For example, a large pool of Usenet newsgroup postings may have colloquialisms, misspellings, and other "words" not found in a dictionary; furthermore, using the most frequent words in such a corpus may allow the attacker to send smaller emails without losing much effectiveness.

3.3 Focused attack

Our second attack—the *focused attack*—assumes knowledge of a specific legitimate email or type of email the attacker wants blocked by the victim's spam filter. This is a Causative Targeted Availability attack. In the focused attack, the attacker sends attack emails to the victim containing words likely to occur in the target email. When SpamBayes trains on this attack email, the spam scores of the targeted tokens increase, so the target message is more likely to be filtered as *spam*. For example, consider a malicious contractor wishing to prevent the victim from receiving messages with competing bids. The attacker sends spam emails to the victim with words such as the names of competing companies, their products, and their

[3]SpamBayes uses at most 150 tokens from E with scores furthest from 0.5 and outside the interval $[0.4, 0.6]$. We call this set $\delta(E)$.

Parameter	Dictionary Attack	Focused Attack	RONI Defense	Threshold Defense
Training set size	2,000, 10,000	5,000	20	2,000, 10,000
Test set size	200, 1,000	N/A	50	200, 1,000
Spam prevalence	0.50, 0.75	0.50	0.50	0.50
Attack fraction	0.001, 0.005, 0.01, 0.02, 0.05, 0.10	0.02 to 0.50 incrementing by 0.02	0.05	0.001, 0.01, 0.05, 0.10
Folds of validation	10	5 repetitions	5 repetitions	5
Target Emails	N/A	20	N/A	N/A

Table 1: Parameters used in our experiments.

employees. The bid messages may even follow a common template, making the attack easier to craft.

The attacker may have different levels of knowledge about the target email. In the extreme case, the attacker might know the exact content of the target email and use all of its words. More realistically, the attacker only guesses a fraction of the email's content. In either case, the attack email may include additional words as well.

The focused attack is more concise than the dictionary attack because the attacker has detailed knowledge of the target and need not affect other messages.

3.4 Optimal attack function

The dictionary and focused attacks can be seen as two instances of a common attack in which the attacker has different amounts of knowledge about the victim's email. Without loss of generality, suppose the attacker generates only a single attack message $\mathbf{a}$. The victim adds it to the training set, trains, and classifies a new message $\mathbf{m}$. Both $\mathbf{a}$ and $\mathbf{m}$ are indicator vectors, where the i^{th} component is true if word i appears in the email. The attacker also has some (perhaps limited) knowledge of the next email the victim will receive. This knowledge can be represented as a distribution $\mathbf{p}$—the vector of probabilities that each word appears in the next message.

The goal of the attacker is to choose an attack email $\mathbf{a}$ that maximizes the *expected spam score* $I_{\mathbf{a}}$ (Equation 3 with the attack message $\mathbf{a}$ in the spam training set) of the next legitimate email $\mathbf{m}$ drawn from distribution $\mathbf{p}$; that is, $\max_{\mathbf{a}} \mathbb{E}_{\mathbf{m}\sim\mathbf{p}} \left[I_{\mathbf{a}}(\mathbf{m})\right]$. In order to describe the optimal attacks under this criterion, we make two observations. First, the spam scores of distinct words do not interact; that is, adding a word w to the attack does not change the score $f(u)$ of some different word $u \neq w$. Second, it can be shown that I is monotonically non-decreasing in each $f(w)$. Therefore the best way to increase $I_{\mathbf{a}}$ is to include additional words in the attack message.

Now let us consider specific choices for the next email's distribution $\mathbf{p}$. First, if the attacker has little knowledge about the words in target emails, we model this by setting $\mathbf{p}$ to be uniform over all vectors $\mathbf{m}$ representing emails. We can optimize the expected spam score by including *all possible words* in the attack email. This *optimal attack* is infeasible in practice but can be simulated: one approximation includes all words in the victim's primary language, such as an English dictionary. This yields the dictionary attack.

Second, if the attacker has specific knowledge of a target email, we can represent this by setting p_i to 1 if and only if the i^{th} word is in the target email. The optimal attack still maximizes the expected spam score, but a more compact attack that is also optimal is to include all of the words in the target email. This is the focused attack.

The attacker's knowledge usually falls between these extremes. For example, the attacker may use information about the distribution of words in English text to make the attack more efficient, such as characteristic vocabulary or jargon typical of the victim. Either of these results in a distribution $\mathbf{p}$ over words in the victim's email. From this it should be possible to derive an optimal constrained attack, but we leave this to future work.

4 Experiments

4.1 Experimental method

In our experiments we use the Text Retrieval Conference (TREC) 2005 spam corpus [4], which is based on the Enron email corpus [11] and contains 92,189 emails (52,790 spam and 39,399 ham). This corpus has several strengths: it comes from a real-world source, it has a large number of emails, and its creators took care that the added spam does not have obvious artifacts to differentiate it. We also use a corpus constructed from a subset of Usenet English postings to generate words for our attacks [18].

We restrict the attacker to have limited control over the headers of attack emails. We implement this assumption either by using the entire header from a randomly selected spam email from TREC (focused attack) or by using an empty header (all other attacks).

We measure the effect of each attack by comparing classification performance of the control and compromised filters using K-fold cross-validation (or K repetitions with new random dataset samples in the case of the focused attack). In cross-validation, we partition the dataset into K subsets and perform K train-test epochs. During the i^{th} epoch, the i^{th} subset is set aside as a test set and the remaining $(K-1)$ subsets are used for training. Each email from our original dataset serves independently as both training and test data.

In the following sections, we show the effect of our attacks on test sets of held-out messages. Because our attacks are designed to cause ham to be misclassified, we only show their effect on ham messages; their effect on spam is marginal. Our graphs do not include error bars since we observed that the variation on our tests was small. See Table 1 for our experimental parameters.

4.2 Dictionary attack results

We examined dictionary attacks as a function of the percent of attack messages in the training set (see Figure 1). It shows the misclassification rates of three dictionary attack variants averaging over ten-fold cross-validation. The optimal attack quickly causes the filter to label all ham emails as *spam*. The Usenet dictionary attack (90,000 top ranked words from the Usenet corpus) causes significantly more ham emails to be misclassified than the Aspell dictionary attack, since it contains common misspellings and slang terms that are not present in the Aspell dictionary (the overlap between the Aspell and Usenet dictionaries is around 61,000 words). These variations of the attack require relatively few attack emails to significantly degrade the SpamBayes accuracy. By 101 attack emails (1% of 10,000), the accuracy falls significantly for each attack variation; at this point most users will gain no advantage from continued use of the filter.

To be fair, although the attack emails make up a small percentage of the *number of messages* in a poisoned inbox, they make up a large percentage of the *number of tokens*. For example, at 204 attack emails (2% of the messages), the Usenet attack includes approximately 6.4 times as many tokens as the original dataset and the Aspell attack includes 7 times. An attack with fewer tokens likely would be harder to detect; however, the number of messages is a more visible feature. It is of significant interest that so few attack messages can degrade a widely-deployed filtering algorithm to such a degree.

4.3 Focused attack results

We run each repetition of the focused attack as follows. First we randomly select a ham email from the TREC corpus to serve as the target of the attack. We use a clean, non-malicious 5,000-message inbox with 50% spam. We

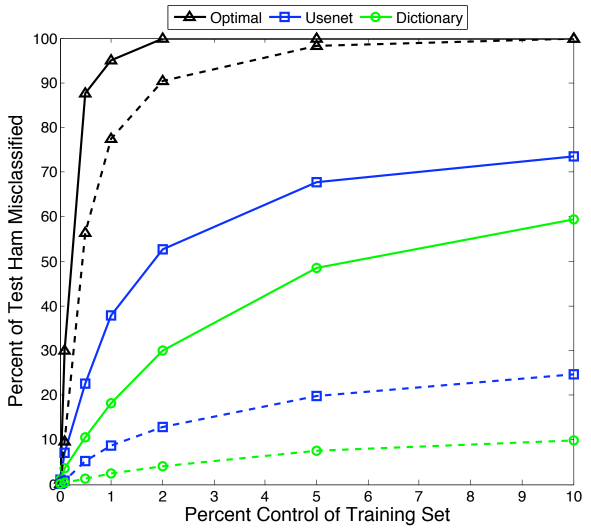

Figure 1: Three dictionary attacks on initial training set of 10,000 messages (50% spam). We plot percent of ham classified as *spam* (dashed lines) and as *spam* or *unsure* (solid lines) against the attack as percent of the training set. We show the optimal attack (black △), the Usenet dictionary attack (blue □), and the Aspell dictionary attack (green ○). Each attack renders the filter unusable with as little as 1% control (101 messages).

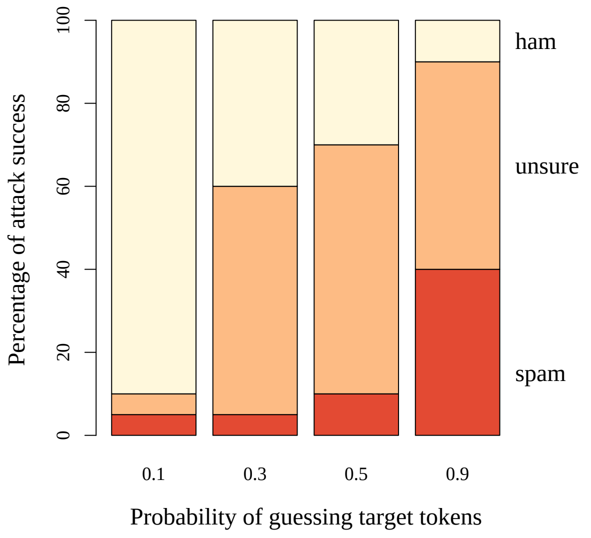

Figure 2: Effect of the targeted attack as a function of the probability of guessing target tokens. Each bar depicts the fraction of target emails classified as *spam*, *ham*, and *unsure* after the attack. The initial inbox contains 5,000 emails (50% spam).

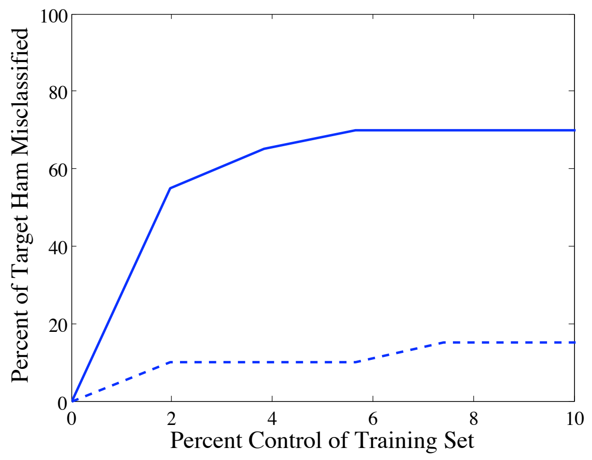

Figure 3: Effect of the focused attack as a function of the number of attack emails with a fixed probability (p=0.5) that the attacker guesses each token. The dashed line shows the percentage of target ham messages misclassified as *spam* after the attack, and the solid line the percentage of targets that are misclassified as *unsure* or *spam* after the attack. The initial inbox contains 5,000 emails (50% spam).

repeat the entire attack procedure independently for 20 randomly-selected target emails.

In Figure 2, we examine the effectiveness of the attack when the attacker has increasing knowledge of the target email by simulating the process of the attacker guessing tokens from the target email. For this figure, there are 300 attack emails—16% of the original number of training emails. We assume that the attacker correctly guesses each word in the target with probability p in $\{0.1, 0.3, 0.5, 0.9\}$—the x-axis of Figure 2. The y-axis shows the proportion of the 20 targets classified as *ham*, *unsure* and *spam*. As expected, the attack is increasingly effective as p increases. If the attacker guesses only 30% of the tokens in the target, classification changes on 60% of the target emails.

In Figure 3, we examine the attack's effect on misclassifications of the targeted emails as the number of attack messages increases. In this figure, we fix the probability of guessing each target token at 0.5. The x-axis is the number of messages in the attack and the y-axis is the percent of messages misclassified. With 100 attack emails, out of a initial mailbox size of 5,000, the target email is misclassified 32% of the time.

We find more insight by examining the attack's effect on three representative emails (see Figure 4). Each of the panels in the figure represents a single target email from each of three attack results: *ham* misclassified as *spam* (Left), *ham* misclassified as *unsure* (Middle), and *ham* correctly classified as *ham* (Right). Each point in the graph represents the before/after score of a token; any point above the line $y = x$ increased due to the attack and any point below is a decrease. From these graphs it is clear that tokens included in the attack typically increase significantly while those not included decrease slightly. Since the increase in score is more significant for included tokens than the decrease in score for excluded tokens, the attack has substantial impact even when the attacker has a low probability of guessing tokens as seen in Figure 2. Further, the before/after histograms in Figure 4 provide a direct indicator of the attack's success.

5 Defenses

5.1 RONI defense

Our Causative attacks work because training on attack emails causes the filter to learn incorrectly and misclassify emails. Each attack email contributes towards the degradation of the filter's performance; if we can measure each email's impact, then we can remove deleterious messages from the training set.

In the *Reject On Negative Impact (RONI) defense*, we measure the incremental effect of each *query email* Q by testing the performance difference with and without that email. We independently sample a 20-message training set T and a 50-message validation set V five times from the initial pool of emails given to SpamBayes for training. We train on both T and $T \cup \{Q\}$ and measure the impact of each query email as the average change in incorrect classifications on V over the five trials. We eliminate Q from training if the impact is significantly negative. We test with 120 random non-attack spam messages and 15 repetitions each of seven variants of the dictionary attacks in Section 3.2.

Preliminary experiments show that the RONI defense is extremely successful against dictionary attacks, identifying 100% of the attack emails without flagging any non-attack emails. Each dictionary attack message causes *at least* an average decrease of 6.8 ham-as-*ham* messages. In sharp contrast, non-attack spam messages cause *at most* an average decrease of 4.4 ham-as-*ham* messages. This clear region of separability means a simple threshold on this statistic is effective at separating dictionary attack emails from non-attack spam.

We plan to extend our initial experiments for the RONI defense with larger test sets along with a larger variation in the number of attack emails. Our initial small experiment, however, gives us confidence that this defense would work given a larger test set due to the large impact a small number of attack emails have on performance.

However, the RONI defense fails to differentiate focused attack emails from non-attack emails. The explanation is simple: the dictionary attack broadly affects

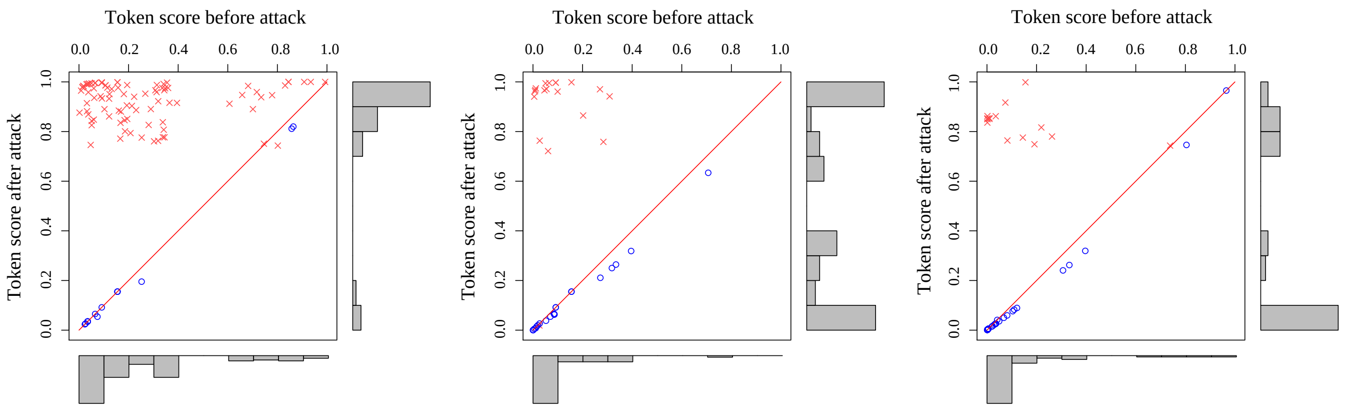

Figure 4: Effect of the focused attack on three representative emails— one graph for each target. Each point is a token in the email. The x-axis is the token spam score in Equation (2) before the attack (0 means ham and 1 means spam). The y-axis is the spam score after the attack. The red $\times$'s are tokens that were included in the attack and the blue $\bigcirc$'s are tokens that were not in the attack. The histograms show the distribution of spam scores before the attack (at bottom) and after the attack (at right).

emails, including training emails, while the focused attack is targeted at a *future* email, so its effects may not be evident on the training set alone.

5.2 Dynamic threshold defense

Distribution-based attacks increase the spam score of ham email but they also tend to increase the spam score of spam. Thus with new θ_0, θ_1 thresholds, it may still be possible to accurately distinguish between these kinds of messages after an attack. Based on this hypothesis, we propose and test a *dynamic threshold defense*, which dynamically adjusts θ_0, θ_1. With an adaptive threshold scheme, attacks that shift all scores will not be effective since rankings are invariant to such shifts.

To determine dynamic values of θ_0 and θ_1, we split the full training set in half. We use one half to train a SpamBayes filter F and the other half as a validation set V. Using F, we obtain a score for each email in V. From this information, we can pick threshold values that more accurately separate ham and spam emails. We define a utility function for choosing threshold t, $g(t) = N_{S,<}(t) \left(N_{S,<}(t) + N_{H,>}(t)\right)^{-1}$, where $N_{S,<}(t)$ is the number of spam emails with scores less than t and $N_{H,>}(t)$ is the number of ham emails with scores greater than t. We select θ_0 so that $g(\theta_0)$ is 0.05 or 0.10, and we select θ_1 so that $g(\theta_1)$ is 0.95 or 0.90, respectively.

This defense shows some promise against the dictionary attacks in a preliminary experiment. As shown in Figure 5, the misclassification of ham emails is significantly reduced compared to SpamBayes alone. At all stages of the attack, ham emails are never classified as *spam* and only a moderate amount of them are labeled as *unsure*. However, while ham messages are often classified properly, the dynamic threshold causes almost all spam messages to be classified as *unsure* even when the attack is only 1% of the inbox. This shows that the dynamic threshold defense failed to adequately separate ham and spam given the number of spam also classified as *unsure*; we are exploring this defense under other choices of the thresholds.

6 Related work

Many authors have examined adversarial learning from a more theoretical perspective. For example, within the Probably Approximately Correct framework, Kearns and Li bound the classification error an adversary that has control over a fraction β of the training set can cause [9]. Dalvi et al. apply game theory to the classification problem [5]. They model interactions between the classifier and attacker as a game and find the optimal counter-strategy for the classifier against an optimal opponent.

In this paper we focus on Causative attacks. Most existing attacks against content-based spam filters are Exploratory attacks that do not influence training but engineer spam messages so they pass through the filter. For example, Lowd and Meek explore reverse-engineering a spam classifier to find high-value messages that the filter does not block [12, 13], Karlberger et al. study the effect of replacing strong spam words with synonyms [8], and Wittel and Wu study the effect of adding common words to spam to get it through a spam filter [19].

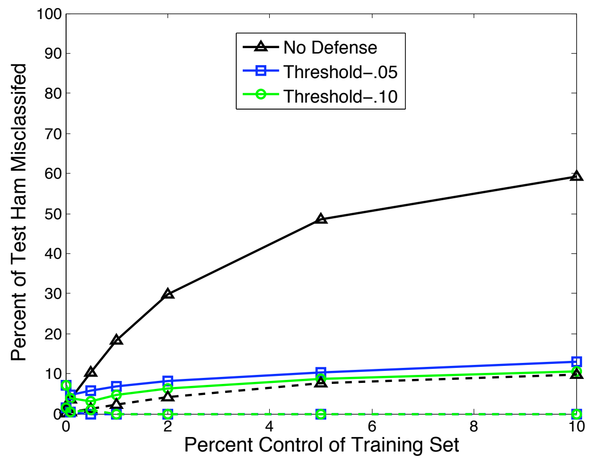

Figure 5: Effect of the threshold defense on the classification of ham messages with the dictionary based attacks. We use a $10,000$ inbox training set of which 50% are spam. The solid lines represent ham messages classified as spam or unsure while the dashed lines show the classification rate of ham messages as spam. Threshold-.05 has a wider range for unsure messages than the Threshold-.10 variation.

Several others have recently developed Causative attacks against machine learning systems. Chung and Mok [2, 3] present a Causative Availability attack against the Autograph worm signature generation system [10], which infers blocking rules based on patterns observed in traffic from suspicious nodes. The main idea is that the attack node first sends traffic that causes Autograph to mark it suspicious, then sends traffic similar to legitimate traffic, resulting in rules that cause denial of service.

Newsome, Karp, and Song [16] present attacks against Polygraph [15], a polymorphic virus detector that uses machine learning. They primarily focus on conjunction learners, presenting Causative Integrity attacks that exploit certain weaknesses not present in other learning algorithms (such as that used by SpamBayes). They also suggest a *correlated outlier attack*, which attacks a naive-Bayes-like learner by adding spurious features to positive training instances, causing the filter to block benign traffic with those features (a Causative Availability attack). They speculate briefly about applying such an attack to spam filters; however, several of their assumptions about the learner are not appropriate in the case of SpamBayes, such as that the learner uses only features indicative of the positive class. Furthermore, although they present insightful analysis, they do not evaluate the correlated outlier attack against a real system. Our attacks use similar ideas, but we develop and test them on a real system. We also explore the value of information to an attacker, and we present and test defenses against the attacks.

7 Conclusion

In this paper, we show that an adversary can effectively disable the SpamBayes spam filter with relatively little system state information and relatively limited control over training data. Our Usenet dictionary attack causes misclassification of 36% of ham messages with only 1% control over the training messages, rendering SpamBayes unusable. Our focused attack changes the classification of the target message 60% of the time with knowledge of only 30% of the target's tokens.

We also explore two successful defenses. The RONI defense filters out dictionary attack messages with complete success. The dynamic threshold defense also mitigates the effect of the dictionary attacks. Focused attacks are especially difficult to defend against because of the attacker's extra knowledge; developing effective defenses is an important open problem. In future work, we intend to continue refining our defenses.

Our attacks and defenses should also apply to other spam filtering systems based on similar learning algorithms, such as BogoFilter and the Bayesian component of SpamAssassin although their effect may vary. Similar techniques may also be effective against other learning systems, such as worm or intrusion detection.

Acknowledgments

We would like to thank Satish Rao, Carla Brodley, and our anonymous reviewers for their useful comments and suggestions on this research.

This work was supported in part by the Team for Research in Ubiquitous Secure Technology (TRUST), which receives support from the National Science Foundation (NSF award #CCF-0424422), the Air Force Office of Scientific Research (AFOSR #FA9550-06-1-0244), Cisco, British Telecom, ESCHER, Hewlett-Packard, IBM, iCAST, Intel, Microsoft, ORNL, Pirelli, Qualcomm, Sun, Symantec, Telecom Italia, and United Technologies; in part by California state Microelectronics Innovation and Computer Research Opportunities grants (MICRO ID#06-148 and #07-012) and Siemens; and in part by the cyber-DEfense Technology Experimental Research laboratory (DETERlab), which receives support from the Department of Homeland Security Homeland Security Advanced Research Projects Agency (HSARPA award #022412) and AFOSR (#FA9550-07-1-0501). The opinions expressed in this paper are solely those of the authors and do not necessarily reflect the opinions of any funding agency, the State of California, or the U.S. government.

References

[1] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Proceedings of the ACM Symposium on InformAtion, Computer, and Communications Security (ASIACCS'06)*, March 2006.

[2] Simon P. Chung and Aloysius K. Mok. Allergy attack against automatic signature generation. In *Recent Advances in Intrusion Detection (RAID)*, pages 61–80, 2006.

[3] Simon P. Chung and Aloysius K. Mok. Advanced allergy attacks: Does a corpus really help? In *Recent Advances in Intrusion Detection (RAID)*, pages 236–255, 2007.

[4] Gordon Cormack and Thomas Lynam. Spam corpus creation for TREC. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS 2005)*, July 2005.

[5] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–108, Seattle, WA, 2004. ACM Press.

[6] Ronald A. Fisher. Question 14: Combining independent tests of significance. *American Statistician*, 2(5):30–30J, 1948.

[7] Paul Graham. A plan for spam. `http://www.paulgraham.com/spam.html`, August 2002.

[8] Christoph Karlberger, Günther Bayler, Christopher Kruegel, and Engin Kirda. Exploiting redundancy in natural language to penetrate Bayesian spam filters. In *WOOT'07: Proceedings of the first conference on First USENIX Workshop on Offensive Technologies*, 2007.

[9] Michael Kearns and Ming Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4):807–837, 1993.

[10] Hyang-Ah Kim and Brad Karp. Autograph: Toward automated, distributed worm signature detection. In *USENIX Security Symposium*, August 2004.

[11] Bryan Klimt and Yiming Yang. Introducing the Enron corpus. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, July 2004.

[12] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 641–647, 2005.

[13] Daniel Lowd and Christopher Meek. Good word attacks on statistical spam filters. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, 2005.

[14] Tony Meyer and Brendon Whateley. SpamBayes: Effective open-source, Bayesian based, email classification system. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, July 2004.

[15] James Newsome, Brad Karp, and Dawn Song. Polygraph: Automatically generating signatures for polymorphic worms. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 226–241, May 2005.

[16] James Newsome, Brad Karp, and Dawn Song. Paragraph: Thwarting signature learning by training maliciously. In *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID 2006)*, September 2006.

[17] Gary Robinson. A statistical approach to the spam problem. *Linux Journal*, March 2003.

[18] Cyrus Shaoul and Chris Westbury. A USENET corpus (2005-2007), October 2007. `http://www.psych.ualberta.ca/~westburylab/downloads/usenetcorpus.download.html`.

[19] Gregory L. Wittel and S. Felix Wu. On attacking statistical spam filters. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, 2004.