Mach Learn (2010) 81: 121–148
DOI 10.1007/s10994-010-5188-5

# The security of machine learning

**Marco Barreno · Blaine Nelson · Anthony D. Joseph · J.D. Tygar**

Received: 8 April 2008 / Revised: 15 April 2010 / Accepted: 15 April 2010 / Published online: 20 May 2010


**Abstract** Machine learning's ability to rapidly evolve to changing and complex situations has helped it become a fundamental tool for computer security. That adaptability is also a vulnerability: attackers can exploit machine learning systems. We present a taxonomy identifying and analyzing attacks against machine learning systems. We show how these classes influence the costs for the attacker and defender, and we give a formal structure defining their interaction. We use our framework to survey and analyze the literature of attacks against machine learning systems. We also illustrate our taxonomy by showing how it can guide attacks against SpamBayes, a popular statistical spam filter. Finally, we discuss how our taxonomy suggests new lines of defenses.

**Keywords** Security · Adversarial learning · Adversarial environments

## 1 Introduction

If we hope to use machine learning as a general tool for computer applications, it is incumbent on us to investigate how well machine learning performs under adversarial conditions. When a learning algorithm performs well in adversarial conditions, we say it is an algorithm for *secure learning*. This raises the natural question: how do we evaluate the quality of a learning system and determine whether it satisfies requirements for secure learning?

Editors: Pavel Laskov and Richard Lippmann.

M. Barreno (✉) · B. Nelson · A.D. Joseph · J.D. Tygar
Computer Science Division, University of California, Berkeley, CA 94720-1776, USA
e-mail: barreno@cs.berkeley.edu

B. Nelson
e-mail: nelsonb@cs.berkeley.edu

A.D. Joseph
e-mail: adj@cs.berkeley.edu

J.D. Tygar
e-mail: tygar@cs.berkeley.edu

Machine learning advocates have proposed learning-based systems for a variety of security applications, including spam detection and network intrusion detection. Their vision is that machine learning will allow a system to respond to evolving real-world inputs, both hostile and benign, and learn to reject undesirable behavior. The danger is that an attacker will attempt to exploit the adaptive aspect of a machine learning system to cause it to fail. Failure consists of causing the learning system to produce errors: if it misidentifies hostile input as benign, hostile input is permitted through the security barrier; if it misidentifies benign input as hostile, desired input is rejected. The adversarial opponent has a powerful weapon: the ability to design training data that will cause the learning system to produce rules that misidentify inputs. If users detect the failure, they may lose confidence in the system and abandon it. If users do not detect the failure, then the risks can be even greater.

It is well established in computer security that evaluating a system involves a continual process of first, determining classes of attacks on the system; second, evaluating the resilience of the system against those attacks; and third, strengthening the system against those classes of attacks. Our paper follows exactly this model in evaluating *secure learning*.

First, we identify different classes of attacks on machine learning systems (Sect. 2). While many researchers have considered particular attacks on machine learning systems, previous research has not presented a comprehensive view of attacks. In particular, we show that there are at least three interesting dimensions to potential attacks against learning systems: (1) they may be *Causative* in that they alter the training process, or they may be *Exploratory* and exploit existing weaknesses; (2) they may be attacks on *Integrity* aimed at *false negatives* (allowing hostile input into a system) or they may be attacks on *Availability* aimed at *false positives* (preventing benign input from entering a system); and (3) they may be *Targeted* at a particular input or they may be *Indiscriminate* in which inputs fail. Each of these dimensions operates independently, so we have at least eight distinct classes of attacks on machine learning system. We can view secure learning as a game between an *attacker* and a *defender*; the taxonomy determines the structure of the game and cost model.

Second, we consider how resilient existing systems are against these attacks (Sect. 3). There has been a rich set of work in recent years on secure learning systems, and we evaluate many attacks against machine learning systems and proposals for making systems secure against attacks. Our analysis describes these attacks in terms of our taxonomy and secure learning game, demonstrating that our framework captures the salient aspects of each attack.

Third, we investigate some potential defenses against these attacks (Sect. 4). Here the work is more tentative, and it is clear that much remains to be done, but we discuss a variety of techniques that show promise for defending against different types of attacks.

Finally, we illustrate our different classes of attacks by considering a contemporary machine learning application, the SpamBayes spam detection system (Sect. 5). We construct realistic, effective attacks by considering different aspects of the threat model according to our taxonomy, and we discuss a defense that mitigates some of the attacks.

This paper provides system designers with a framework for evaluating machine learning systems for security applications (illustrated with our evaluation of SpamBayes) and suggests directions for developing highly robust secure learning systems. Our research not only proposes a common language for thinking and writing about secure learning, but goes beyond that to show how our framework works, both in algorithm design and in real system evaluation. We present an essential first step if machine learning is to reach its potential as a tool for use in real systems in potentially adversarial environments.

### 1.1 Notation

We focus on binary classification for security applications, in which a *defender* attempts to separate *instances* of input (data points), some or all of which come from a malicious *attacker*, into harmful and benign classes. This setting covers many interesting security applications, such as host and network intrusion detection, virus and worm detection, and spam filtering. In detecting malicious activity, the *positive* class (label 1) indicates malicious *intrusion* instances while the *negative* class (label 0) indicates benign *normal* instances. A classification error is a *false positive* (*FP*) if a normal instance is classified as positive and a *false negative* (*FN*) if an intrusion instance is classified as negative.

It may be interesting as well to consider cases where a classifier has more than two classes, or even a real-valued output. Indeed, the spam filter SpamBayes, which we consider in our experiments in Sect. 5, uses three labels so it can explicitly label some messages *unsure*. However, generalizing the analysis of errors to more than two classes is not straightforward, and furthermore most systems in practice make a single fundamental distinction (for example, spam messages that the user will never see vs. non-spam and unsure messages that the user will see). For these reasons, and in keeping with common practice in the literature, we limit our analysis to binary classification and leave extension to the multi-class or real-valued cases as future work.

In the *supervised classification* problem, the learner trains on a dataset of $N$ instances, $\mathbf{X} = \{(x, y) \mid x \in \mathcal{X}, y \in \mathcal{Y}\}^N$, given an instance space $\mathcal{X}$ and the label space $\mathcal{Y} = \{0, 1\}$. Given some hypothesis class $\Omega$, the goal is to learn a classification hypothesis (classifier) $f^* \in \Omega$ to minimize errors when predicting labels for new data, or if our model includes a cost function over errors, to minimize the total cost of errors. The cost function assigns a numeric cost to each combination of data instance, true label, and classifier label. The defender chooses a *procedure* $H$, or learning algorithm, for selecting hypotheses. The classifier may periodically interleave *training* steps with the *evaluation*, retraining on some or all of the accumulated old and new data. In adversarial environments, the attacker controls some of the data, which may be used for training. We assume that the learner has some way to get the true labels for its training data and for the purpose of computing cost; the true label might come from manual classification of a training set or from observing the effect of instances on a test system, for example.

The procedure can be any method of selecting a hypothesis; in statistical machine learning, a common procedure is (*regularized*) *empirical risk minimization*. This procedure is an optimization problem where the objective function has an *empirical risk* term and a *regularization* term. Since true cost is often not representable precisely and efficiently, we calculate risk as the expected *loss* given by a *loss function* $\ell$ that approximates true cost; the regularization term $\rho$ captures some notion of hypothesis complexity to prevent *overfitting* the training data, using a weight $\lambda$ to quantify the trade-off. This procedure finds the hypothesis minimizing:

$$f^* = \operatorname*{argmin}_{f \in \Omega} \sum_{(x,y) \in \mathbf{X}} \ell(y, f(x)) + \lambda \rho(f) \tag{1}$$

Many learning methods make a *stationarity* assumption: training data and evaluation data are drawn from the same distribution. This assumption allows us to minimize the risk on the training set as a surrogate for risk on the evaluation data, since evaluation data are not known at training time. However, real-world sources of data often are not stationary and, even worse, attackers can easily break the stationarity assumption with some control of

**Table 1** Notation in this paper

| | |
|---|---|
| $\mathcal{X}$ | Space of data instances |
| $\mathcal{Y}$ | Space of data labels; for classification $\mathcal{Y} = \{0, 1\}$ |
| $\mathfrak{D}$ | Space of distributions over $(\mathcal{X} \times \mathcal{Y})$ |
| $\Omega$ | Space of hypotheses $f : \mathcal{X} \mapsto \mathcal{Y}$ |
| $\mathbb{P}_{\mathrm{T}} \in \mathfrak{D}$ | Training distribution |
| $\mathbb{P}_{\mathrm{E}} \in \mathfrak{D}$ | Evaluation distribution |
| $\mathbb{P} \in \mathfrak{D}$ | Distribution for training and evaluation (Sect. 4.2.3) |
| $x \in \mathcal{X}$ | Data instance |
| $y \in \mathcal{Y}$ | Data label |
| $\mathbf{X}, \mathbf{E}, \mathbf{Z}, \mathbf{C}, \mathbf{T}_i, \mathbf{Q}_i \in (\mathcal{X} \times \mathcal{Y})^N$ | Datasets |
| $H : (\mathcal{X} \times \mathcal{Y})^N \mapsto \Omega$ | Procedure for selecting hypothesis |
| $A_T, A_E : \mathcal{X}^N \times \Omega \mapsto \mathfrak{D}$ | Procedures for selecting distribution |
| $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^{0+}$ | Loss function |
| $C : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ | Cost function |
| $f : \mathcal{X} \mapsto \mathcal{Y}$ | Hypothesis (classifier) |
| $f^* : \mathcal{X} \mapsto \mathcal{Y}$ | Best hypothesis |
| $N$ | Number of data points |
| $K$ | Number of repetitions of a game |
| $M$ | Number of experts (Sect. 4.2.3) |
| $\lambda$ | Trade-off parameter for regularized risk minimization |

either training or evaluation instances. Analyzing and strengthening learning methods in the face of a broken stationarity assumption is the crux of the *secure learning* problem.

We model attacks on machine learning systems as a game between two players, the *attacker* and the *defender*. The game consists of a series of *moves*, or *steps*. Each move encapsulates a choice by one of the players: the attacker alters or selects data; the defender chooses a training procedure for selecting the classification hypothesis.

Table 1 summarizes the notation we use in this paper.

## 2 Framework

### 2.1 Security analysis

Properly analyzing the security of a system requires identifying *security goals* and a *threat model*. Security is concerned with protecting assets from attackers. A security goal is a requirement that, if violated, results in the partial or total compromise of an asset. A threat model is a profile of attackers, describing motivation and capabilities. Here we analyze the security goals and threat model for machine learning systems.

Classifiers are used to make distinctions that advance security goals. For example, a *virus detection system* has the goal of reducing susceptibility to virus infection, either by detecting the virus in transit prior to infection or by detecting an extant infection to expunge. Another example is an *intrusion detection system* (*IDS*), which identifies compromised systems, usually by detecting malicious traffic to and from the system or by detecting suspicious

behavior in the system. A closely related concept is the *intrusion prevention system* (*IPS*), which detects intrusion attempts and then acts to prevent them from succeeding.

In this section we describe security goals and a threat model that are specific to machine learning systems.

### *2.1.1 Security goals*

In a security context the classifier's purpose is to classify malicious events and prevent them from interfering with system operations. We split this general learning goal into two goals:

– *Integrity goal*: To prevent attackers from reaching system assets.
– *Availability goal*: To prevent attackers from interfering with normal operation.

There is a clear connection between false negatives and violation of the integrity goal: malicious instances that pass through the classifier can wreak havoc. Likewise, false positives tend to violate the availability goal because the learner itself denies benign instances.

### *2.1.2 Threat model*

*Attacker goal/incentives* In general the attacker wants to access system assets (typically with false negatives) or deny normal operation (usually with false positives). For example, a virus author wants viruses to pass through the filter and take control of the protected system (a false negative). On the other hand, an unscrupulous merchant may want sales traffic to a competitor's web store to be blocked as intrusions (false positives).

We assume that the attacker and defender each have a *cost function* that assigns a cost to each labeling for any given instance. Cost can be positive or negative; a negative cost is a benefit. It is usually the case that low cost for the attacker parallels high cost for the defender and vice-versa; the attacker and defender would not be adversaries if their goals were aligned. In this paper, unless otherwise stated, for ease of exposition we assume that every cost for the defender corresponds to a similar benefit for the attacker and vice-versa. This assumption is not essential to our arguments, which extend easily to arbitrary cost functions. We take the defender's point of view, so we use "high-cost" to mean high positive cost for the defender.

*Attacker capabilities* We assume that the attacker has knowledge of the training algorithm, and in many cases partial or complete information about the training set, such as its distribution. For example, the attacker may have the ability to eavesdrop on all network traffic over the period of time in which the learner gathers training data. The attacker may be able to modify or generate data used in training; we consider cases in which the attacker can and cannot control some of the learner's training data. While we think that this is the most accurate assumption for most cases, if we do err, we wish to follow the common practice in computer security research of erring on the side of overestimating the attacker's capabilities rather than underestimating them.

In general we assume the attacker can generate arbitrary instances; however, many settings do impose significant restrictions on the instances generated by the attacker. For example, when the learner trains on data from the attacker, sometimes it is safe to assume that the attacker cannot choose the label for training, such as when training data is carefully hand labeled. As another example, an attacker may have complete control over data packets being sent from the attack source, but routers in transit may add to or alter the packets as well as affect their timing and arrival order.

When the attacker controls training data, an important limitation to consider is what fraction of the training data the attacker can control and to what extent. If the attacker has

arbitrary control over 100% of the training data, it is difficult to see how the learner can learn anything useful; however, even in such cases there are learning strategies that can make the attacker's task more difficult (see Sect. 4.2.3). We primarily examine intermediate cases and explore how much influence is required for the attacker to defeat the learning procedure.

## 2.2 Taxonomy

We present a taxonomy categorizing attacks against learning systems along three axes:

INFLUENCE

- *Causative* attacks influence learning with control over training data.
- *Exploratory* attacks exploit misclassifications but do not affect training.

SECURITY VIOLATION

- *Integrity* attacks compromise assets via false negatives.
- *Availability* attacks cause denial of service, usually via false positives.

SPECIFICITY

- *Targeted* attacks focus on a particular instance.
- *Indiscriminate* attacks encompass a wide class of instances.

The first axis 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 instances to the classifier and possibly observe its decisions on these carefully crafted instances (an *Exploratory* attack).

The second axis indicates the type of security violation the attacker causes: either (a) allowing harmful instances to slip through the filter as false negatives (an *Integrity* violation); or (b) creating a denial of service in which benign instances are incorrectly filtered as false positives (an *Availability* violation).

The third axis refers to how specific the attacker's intention is: whether (a) the attack is highly *Targeted* to degrade the classifier's performance on one particular instance or (b) the attack aims to cause the classifier to fail in an *Indiscriminate* fashion on a broad class of instances. Each axis, especially this one, is actually a spectrum of choices.

A preliminary version of this taxonomy appears in previous work (Barreno et al. 2006). Here we extend the framework and show how the taxonomy shapes the game played by the attacker and defender (see Sect. 2.4). The INFLUENCE axis of the taxonomy determines the structure of the game and the move sequence. The SPECIFICITY and SECURITY VIOLATION axes of the taxonomy determine the general shape of the cost function: an *Integrity* attack benefits the attacker on false negatives, and therefore focuses high cost (to the defender) on false negatives, and an *Availability* attack focuses high cost on false positives; a *Targeted* attack focuses high cost only on a small number of instances, while an *Indiscriminate* attack spreads high cost over a broad range of instances.

## 2.3 Examples

Here we give four hypothetical attack scenarios, each with two variants, against a token-based spam filter that uses machine learning, such as the SpamBayes filter discussed in Sect. 5. Table 2 summarizes the taxonomy and shows where these examples fit within it.

**Table 2** Our taxonomy of attacks against machine learning systems, with examples from Sect. 2.3

| | Integrity | Availability |
|---|---|---|
| Causative | | |
| Targeted | *The spam foretold*: mis-train a particular spam | *The rogue filter*: mis-train filter to block a certain message |
| Indiscriminate | *The spam foretold*: mis-train any of several spams | *The rogue filter*: mis-train filter to broadly block normal email |
| Exploratory | | |
| Targeted | *The shifty spammer*: obfuscate a chosen spam | *The unwanted reply*: flood a particular target inbox |
| Indiscriminate | *The shifty spammer*: obfuscate any spam | *The unwanted reply*: flood any of several target inboxes |

This section is meant to give the reader an intuition for how the taxonomy organizes attacks against machine learning systems. There are many practical considerations that may make attacks difficult. In Sect. 3 we discuss some concrete attacks that have been published in the literature, which more fully address relevant practical concerns (see Sect. 3.5 for a summary).

### *2.3.1 Causative Integrity attack: The spam foretold*

In a *Causative Integrity* attack, the attacker uses control over training to cause spam to slip past the classifier as false negatives.

Example: an attacker wants the defender's spam filter not to flag a novel spam message. The defender re-trains periodically on new messages, so the attacker sends non-intrusion traffic that is carefully chosen to resemble the desired spam, such as by including many of the spam's tokens re-ordered into a benign message, to mis-train the learner to fail to block the eventual spam campaign. As a trivial example, the spam sales pitch "What watch do you want? Really, buy it now!" could be recast as "Watch what you buy now! Do you really want it?"; these excerpts have different semantic meanings but appear identical to a unigram spam filter. The primary limitation of this attack is ensuring that the filter is retrained on the attack messages before the campaign commences. The attack's success would also depend on the particular words in the spam—messages that contain words highly indicative of spam, such as 'Viagra,' may be more difficult to mis-train.

This example might be *Targeted* if the attacker already has a particular spam to send and needs to cause the learner to miss that particular message. It might be *Indiscriminate*, on the other hand, if the attacker has a polymorphic spam and could use any of a large number of variants of it for the campaign, in which case the attack need only fool the learner on any one of these variations.

### *2.3.2 Causative Availability attack: The rogue filter*

In a *Causative Availability* attack, the attacker uses control over training instances to interfere with operation of the mailing system, such as by blocking legitimate email.

Example: an attacker wants normal email to be classified as spam so the intended recipient doesn't receive them. The attacker generates and sends attack messages that resemble

benign messages when the defender is collecting training data to train the spam filter. The attack messages include both a spam component and benign words, which subsequently also become erroneously associated with spam. When the learner trains on the attack data, the spam filter will start to filter normal messages as spam. The primary impediment to this attack is for the attacker to ensure their messages are used to train the filter, though in many cases this is realistic since all available messages are used for training.

We more fully explore an attack of this type in Sect. 5.

This attack could be *Targeted* to block a particular message (see focused attacks in Sect. 5). On the other hand, it might be *Indiscriminate* and attempt to block a significant portion of all legitimate messages (see dictionary attacks in Sect. 5).

#### *2.3.3 Exploratory Integrity attack: The shifty spammer*

In an *Exploratory Integrity* attack, the attacker crafts spam so as to evade the classifier without direct influence over the classifier itself.

Example: an attacker modifies and obfuscates spam, such as by changing headers or by exchanging common spam words for less common synonyms. If successful, these modifications prevent the filter from recognizing the altered spam as malicious, so they are placed into the user's inbox (see Sect. 3.3 for additional discussion of these attacks that use good words to sanitize spam). A practical consideration for this attack is whether the attacker has a feedback mechanism to observe the success of the evasion. Also, the obfuscated message may have less value to the spammer (it may be less effective at generating sales), so there might be a limit on the extent of obfuscation that is practical.

In the *Targeted* version of this attack, the attacker has a particular spam to get past the filter. In the *Indiscriminate* version, the attacker has no particular preference and can search for any spam that succeeds, such as by modifying a number of different spam campaigns to see which modifications evade the filter (of course, in this case the attacker must take care not to appear anomalous simply due to the large number of exploratory instances).

#### *2.3.4 Exploratory Availability attack: The mistaken identity*

In an *Exploratory Availability* attack, the attacker interferes with legitimate operation without influence over training.

Example: an attacker wishes to interfere with the target's ability to read legitimate email messages. The attacker creates a spam campaign in which the target's email address appears as the From: address of the spam messages. The target will receive a flood of spurious, non-spam messages such as bounces for nonexistent addresses, vacation replies, and angry responses demanding the spam to stop. For a large enough spam campaign, these messages will completely overwhelm the target's inbox, making the task of finding the legitimate messages among them so costly as to be impractical. This attack is within the means of most large-scale spammers; however, since it requires a large number of messages to use the same From: address, the number of targets may not be very high. Furthermore, some spam filters now block many bounce messages, so the volume of the spam campaign may need to be high enough to overwhelm the target solely with vacation responses and hand-written replies.

It does not make sense to consider this attack targeted to specific messages, since it necessarily affects the entire inbox; however, the attacker might target certain people or a broader set. In the *Targeted* version, the attacker has a particular inbox to flood with replies. In the *Indiscriminate* version, the attacker has a broader set of inboxes to choose among, such as when going after an organization rather than a single individual.

### 2.4 The adversarial learning game

This section models attacks on learning systems as games where moves represent strategic choices. The choices and computations in a move depend on information produced by previous moves (when a game is repeated, this includes previous iterations). In an *Exploratory* attack, the attacker chooses a procedure $A_E$ that affects the evaluation data $\mathbf{E}$, and in a *Causative* attack, the attacker also chooses a procedure $A_T$ to manipulate the training data $\mathbf{X}$. The defender chooses a learning algorithm $H$. This formulation gives us a theoretical basis for analyzing the interactions between attacker and defender. Refer back to Table 1 for a summary of notation.

The choices made by the attacker and defender offer a variety of different domain-specific strategies for both players. In Sects. 3 and 4, we discuss how attack and defense strategies have been developed in different domains, and we highlight important aspects of the game in those settings. In Sects. 3.1 and 3.2, we discuss practical examples of the attacker's choices in the *Causative* game, and we discuss the defender's choice in Sect. 4.2. Similarly, for an *Exploratory* attack, we discuss realistic instances of the attacker's choice for $A_E$ in Sects. 3.3 and 3.4, and we discuss the defender's choice for an algorithm $H$ in Sect. 4.1.

#### *2.4.1 Exploratory game*

First we present the formal version of the game for *Exploratory* attacks, and then we explain it in greater detail:

1. **Defender** Choose procedure $H$ for selecting hypothesis
2. **Attacker** Choose procedure $A_E$ for selecting distribution
3. Evaluation:
   - Reveal distribution $\mathbb{P}_{\mathrm{T}}$
   - Sample dataset $\mathbf{X}$ from $\mathbb{P}_{\mathrm{T}}$
   - Compute $f \leftarrow H(\mathbf{X})$
   - Compute $\mathbb{P}_{\mathrm{E}} \leftarrow A_E(\mathbf{X}, f)$
   - Sample dataset $\mathbf{E}$ from $\mathbb{P}_{\mathrm{E}}$
   - Assess total cost: $\sum_{(x,y)\in\mathbf{E}} C(x, f(x), y)$

The defender's move is to choose a learning algorithm (procedure) $H$ for creating hypotheses from datasets. Many procedures used in machine learning have the form of (1). For example, the defender may choose a *support vector machine* (SVM) with a particular kernel, loss, regularization, and cross-validation plan. The attacker's move is then to choose a procedure $A_E$ to produce a distribution on which to evaluate the hypothesis that $H$ generates. (The degree of control the attacker has in generating the dataset and the degree of information about $\mathbf{X}$ and $f$ that $A_E$ has access to are setting-specific.)

After the defender and attacker have both made their choices, the game is evaluated. A training dataset $\mathbf{X}$ is drawn from some fixed and possibly unknown distribution $\mathbb{P}_{\mathrm{T}}$, and training produces $f = H(\mathbf{X})$. The attacker's procedure $A_E$ produces distribution $\mathbb{P}_{\mathrm{E}}$, which is based in general on $\mathbf{X}$ and $f$, and an evaluation dataset $\mathbf{E}$ is drawn from $\mathbb{P}_{\mathrm{E}}$. Finally, the attacker and defender incur cost based on the performance of $f$ evaluated on $\mathbf{E}$.

The procedure $A_E$ generally depends on $\mathbf{X}$ and $f$, but the amount of information an attacker actually has is setting specific (in the least restrictive case the attacker knows $\mathbf{X}$ and $f$ completely). The attacker may know a subset of $\mathbf{X}$ or the family $\Omega$ of $f$. However, the procedure $A_E$ may also involve acquiring information dynamically. For instance, in many cases, the procedure $A_E$ can *query* the classifier, treating it as an oracle that provides labels

for query instances; this is one particular degree of information that $A_E$ can have about $f$. Attacks that use this technique are *probing attacks*. Probing can reveal information about the classifier. On the other hand, with sufficient prior knowledge about the training data and algorithm, the attacker may be able to find high-cost instances without probing.

#### 2.4.2 Causative game

The game for *Causative* attacks is similar:

1. **Defender** Choose procedure $H$ for selecting hypothesis
2. **Attacker** Choose procedures $A_T$ and $A_E$ for selecting distributions
3. Evaluation:
   - Compute $\mathbb{P}_{\mathrm{T}} \leftarrow A_T$
   - Sample dataset $\mathbf{X}$ from $\mathbb{P}_{\mathrm{T}}$
   - Compute $f \leftarrow H(\mathbf{X})$
   - Compute $\mathbb{P}_{\mathrm{E}} \leftarrow A_E(\mathbf{X}, f)$
   - Sample dataset $\mathbf{E}$ from $\mathbb{P}_{\mathrm{E}}$
   - Assess total cost: $\sum_{(x,y)\in\mathbf{E}} C(x, f(x), y)$

This game is very similar to the *Exploratory* game, but the attacker can choose $A_T$ to affect the training data $\mathbf{X}$. The attacker may have various types of influence over the data, ranging from arbitrary control over some fraction of instances to a small biasing influence on some aspect of data production; details depend on the setting.

Control over data used for training opens up new strategies to the attacker. Cost is based on the interaction of $f$ and $\mathbf{E}$. In the *Exploratory* game the attacker chooses $\mathbf{E}$ while the defender controls $f$; in the *Causative* game the attacker also has influence on $f$. With this influence, the attacker can proactively cause the learner to produce bad classifiers.

#### 2.4.3 Iteration

We have analyzed these games as *one-shot games*, in which players minimize cost when each move happens only once. We can also consider an *iterated game*, in which the game repeats several times and players minimize total accumulated cost. In this setting, we assume players have access to all information from previous iterations of the game.

## 3 Attacks: categorizing related work

This section surveys examples of learning in adversarial environments from the literature. Our taxonomy provides a basis for evaluating the resilience of the systems described, analyzing the attacks against them in preparation for constructing defenses.

### 3.1 Causative Integrity attacks

*Contamination in PAC learning* Kearns and Li (1993) extend Valiant's *probably approximately correct* (PAC) learning framework (Valiant 1984, 1985) to prove bounds for maliciously chosen errors in the training data. In PAC learning, an algorithm succeeds if it can, with probability at least $1 - \delta$, learn a hypothesis that has at most probability $\varepsilon$ of making an incorrect prediction on an example drawn from the same distribution. Kearns and Li examine the case where an attacker has arbitrary control over some fraction $\beta$ of the training

**Table 3** Related work in the taxonomy

| | Integrity | Availability |
|---|---|---|
| Causative | | |
| Targeted | Kearns and Li (1993), Newsome et al. (2006) | Kearns and Li (1993), Newsome et al. (2006), Chung and Mok (2007), Nelson et al. (2008) |
| Indiscriminate | Kearns and Li (1993), Newsome et al. (2006) | Kearns and Li (1993), Newsome et al. (2006), Chung and Mok (2007), Nelson et al. (2008) |
| Exploratory | | |
| Targeted | Tan et al. (2002), Lowd and Meek (2005b), Wittel and Wu (2004), Lowd and Meek (2005a) | Moore et al. (2006) |
| Indiscriminate | Fogla and Lee (2006), Lowd and Meek (2005b), Wittel and Wu (2004) | Moore et al. (2006) |

examples (this specifies the form that $A_T$ takes in our *Causative* game). They prove that in general the attacker can prevent the learner from succeeding if $\beta \geq \varepsilon/(1+\varepsilon)$, and for some classes of learners they show this bound is tight.

This work provides an interesting and useful bound on the ability to succeed at PAC-learning. The analysis broadly concerns both *Integrity* and *Availability* attacks as well as both *Targeted* and *Indiscriminate*. However, not all learning systems fall into the PAC-learning model.

*Red herring attack* Newsome et al. (2006) present *Causative Integrity* and *Causative Availability* attacks against Polygraph (Newsome et al. 2005), a polymorphic virus detector that learns virus signatures using both a conjunction learner and a naive-Bayes-like learner. They present *red herring* attacks against conjunction learners that exploit certain weaknesses not present in other learning algorithms (these are *Causative Integrity* attacks, both *Targeted* and *Indiscriminate*).

The idea is that the attacker chooses $\mathbb{P}_{\mathrm{T}}$ to introduce spurious features into all malicious instances that the defender uses for training. The malicious instances produced by $\mathbb{P}_{\mathrm{E}}$, however, lack the spurious features and therefore bypass the filter, which erroneously generalized that the spurious features were necessary elements of the malicious behavior.

### 3.2 Causative Availability attacks

*Correlated outlier attack* Newsome et al. (2006) 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 (an *Availability* attack).

As with the red herring attacks, these correlated outlier attacks fit neatly into our causative game; this time $\mathbb{P}_{\mathrm{T}}$ includes spurious features in malicious instances, causing $H$ to produce an $f$ that classifies many benign instances as malicious.

*Allergy attack* Chung and Mok (2006, 2007) present *Causative Availability* attacks against the Autograph worm signature generation system (Kim and Karp 2004). Autograph operates

in two phases. First, it identifies infected nodes based on behavioral patterns, in particular scanning behavior. Second, it observes traffic from the identified nodes and infers blocking rules based on observed patterns. Chung and Mok describe an attack that targets traffic to a particular resource. In the first phase, an attack node convinces Autograph that it is infected by scanning the network. In the second phase, the attack node sends crafted packets mimicking targeted traffic, causing Autograph to learn rules that block legitimate access and create a denial of service.

In the context of our causative game, the attacker's choice of $\mathbb{P}_{\mathrm{T}}$ provides the traffic for both phases of Autograph's learning. When Autograph produces a hypothesis $f$ that depends on the carefully crafted traffic from the attacker, it will block access to legitimate traffic from $\mathbb{P}_{\mathrm{E}}$ that shares patterns with the malicious traffic.

*Attacking SpamBayes* Nelson et al. (2008) demonstrate *Causative Availability* attacks (both *Targeted* and *Indiscriminate*) against the SpamBayes statistical spam classifier. We examine these attacks in depth in Sect. 5.

### 3.3 Exploratory Integrity attacks

Some *Exploratory Integrity* attacks mimic statistical properties of the normal traffic to camouflage intrusions. In the *Exploratory* game, the attacker's move produces instances $\mathbf{E}$ that statistically resemble normal traffic in the training data $\mathbf{X}$ as measured by the learning procedure $H$.

*Polymorphic blending attack* *Polymorphic blending attacks* encrypt attack traffic in such a way that it appears statistically identical to normal traffic. Fogla and Lee (2006) present a formalism for reasoning about and generating *polymorphic blending attack* instances to evade intrusion detection systems.

*Attacking a sequence-based IDS* Tan et al. (2002) describe a mimicry attack against the `stide` anomaly-based intrusion detection system (IDS). They modify exploits of the `passwd` and `traceroute` programs to accomplish the same ends using different sequences of system calls: the shortest subsequence in attack traffic that does not appear in normal traffic is longer than the IDS window size, evading detection. In subsequent work Tan et al. (2003) characterize their attacks as part of a larger class of information hiding techniques which they demonstrate can make exploits mimic either normal call sequences or the call sequence of another less severe exploit.

Independently, Wagner and Soto (2002) have also developed mimicry attacks against a sequence-based IPS called pH.[1] Using the machinery of finite automata, they construct a framework for testing whether an IDS is susceptible to mimicry for a particular exploit. In doing so, they develop a tool for validating IDSs on a wide-range of variants of a particular attack and suggest that similar tools should be more broadly employed to identify the vulnerabilities of an IDS.

Overall, these mimicry attacks against sequence-based anomaly detection systems underscore critical weaknesses in these systems that allow attackers to obfuscate critical elements of their exploits to avoid detection. Further they highlight how an IDS may appear to perform well against a known exploit but, unless it captures necessary elements of the intrusion, the exploit can easily be adapted to circumvent the detector. See Sect. 4.1.1 for more discussion.

[1] Wagner and Soto (2002) treat pH as an IDS rather than an IPS, though that distinction is not relevant to the attacks they present.

*Good word attacks* Several authors demonstrate *Exploratory integrity* attacks using similar principles against spam filters. Lowd and Meek (2005b) and Wittel and Wu (2004) develop attacks against statistical spam filters that add *good words*, or words the filter considers indicative of non-spam, to spam emails. This type of modification can make spam emails appear innocuous to the filter, especially if the words are chosen to be ones that appear often in non-spam email and rarely in spam email.

*Reverse engineering classifiers* Lowd and Meek (2005a) approach the *Exploratory Integrity* attack problem from a different angle: they give an algorithm for an attacker to reverse engineer a classifier. The attacker seeks the highest cost (lowest cost for the attacker) instance that the classifier labels *negative*. This work is interesting in its use of a cost function over instances for the attacker rather than simple positive/negative classification. We explore this work in more detail in Sect. 4.1.2.

### 3.4 Exploratory Availability attacks

*Exploratory Availability* attacks against non-learning systems abound in the literature: almost any *denial of service* (DoS) attack falls into this category, such as those described by Moore et al. (2006).

However, *Exploratory Availability* attacks against the learning components of systems are not common. We suspect this is because if an attacker wishes to cause denial of service but has no control over the learning process, other avenues of attack are more fruitful.

Here is one hypothetical scenario: if a learning IPS has trained on intrusion traffic and has the policy of blocking hosts that originate intrusions, an attacker could send intrusions that appear to originate from a legitimate host, convincing the IPS to block that host. Another possibility is to take advantage of a computationally expensive learning component: for example, spam filters that use image processing to detect advertisements in graphical attachments can take significantly more time than text-based filtering (Dredze et al. 2007; Wang et al. 2007). An attacker could exploit such overhead by sending many emails with images, causing the expensive processing to delay and perhaps even block messages.

### 3.5 Practical considerations for attacks

The attacks described in this section highlight several practical considerations that must be overcome to design an effective attack against a machine learning system. In many cases, realistic attacks on an IPS need to be valid executables—we discuss several such attacks that create valid sequences of system calls in Sect. 3.3 (e.g. attacks by Tan et al. 2002). Other attacks require the attacker to spoof normal behavior—features such as the address of the target system may be difficult to spoof, but Chung and Mok (2007) demonstrate that certain components of HTTP requests sometimes used as features for detection can easily be spoofed. Mahoney and Chan (2003) and Chung and Mok (2007) suggest that poor choices of features can facilitate attacks on an IPS, which we discuss further in Sect. 4.1.1. And finally, Fogla and Lee (2006) discuss in depth the opportunities and constraints of generating attack traffic that appears statistically identical to benign traffic, as mentioned in Sect. 3.3.

## 4 Defenses

This section gives an overview of possible defenses against the types of attacks introduced in previous sections. We discuss both defense mechanisms from prior work as well as ideas

for new defenses. The game between attacker and defender and the taxonomy that we introduce in Sect. 3 provide a foundation on which to construct defense strategies against broad classes of attacks. We address *Exploratory* and *Causative* attacks separately; in Sect. 4.2.3 we discuss the broader setting of an iterated game.

In all cases, defenses present a trade-off: changing the algorithms to make them more robust against (worst-case) attacks will generally make them *less* effective on average. Analyzing this trade-off is an important part of developing defenses.

## 4.1 Defending against Exploratory attacks

*Exploratory* attacks do not corrupt the training data but attempt to find vulnerabilities in the learned hypothesis. When producing the evaluation distribution, the attacker attempts to construct an *unfavorable evaluation distribution* concentrating probability mass on high-cost instances; in other words, the attacker's procedure $A_E$ tries to construct an evaluation distribution $\mathbb{P}_{\mathrm{E}}$ on which the learner predicts poorly (violating stationarity) and the cost computed in the last step of the *Exploratory* game is high. This section examines defender strategies that make it difficult for the attacker to construct such a distribution.

In the *Exploratory* game, the defender makes a move before observing contaminated data. The defender can impede the attacker's ability to reverse engineer the classifier by limiting access to information about the training procedure and data. With less information, $A_E$ has difficulty producing an unfavorable evaluation distribution. Nonetheless, even with incomplete information, the attacker may be able to construct an unfavorable evaluation distribution using a combination of *prior knowledge* and *probing*.

### *4.1.1 Defenses against attacks without probing*

Part of our security analysis involves identifying aspects of the system that should be kept secret. In securing a learner, we limit information to make it difficult for an attacker to conduct an attack.

*Training data* Preventing the attacker from knowing the training data limits the attacker's ability to reconstruct internal states of the classifier. There is a tension between collecting training data that fairly represents the real world instances and keeping all aspects of that data secret. In most situations, it is difficult to use completely secret training data, though the attacker may have only partial information about its distribution.

*Feature selection* We can harden classifiers against attack through attention to features in the feature selection and learning steps (which are both internal steps of the defender's hypothesis selection procedure $H$). Feature selection is the process of choosing a *feature map* that transforms raw measurements into the feature space used by the learning algorithm. In the learning step, the learning algorithm builds its model or signature using particular features from the map's feature space; this *choice of features* for the model or signature is also sometimes referred to as feature selection, though we consider it to be part of the learning process, after the feature map has been established. For example, one feature map for email message bodies might transform each token to a Boolean feature indicating its presence; another map might specify a real-valued feature indicating the relative frequency of each word in the message compared to its frequency in natural language; yet another map might count sequences of $n$ characters and specify an integer feature for each character $n$-gram indicating how many times it appears. In each of these cases, a learner will construct a model

or signature that uses certain features (tokens present or absent; relative frequency of words present; character $n$-gram counts) to decide whether an instance is benign or malicious.

Obfuscation of spam-indicating words (an attack on the feature set) is a common *Targeted Exploratory Integrity* attack. Sculley et al. (2006) use inexact string matching in feature selection to defeat obfuscations of words in spam emails. They choose a feature map based on character subsequences that are robust to character addition, deletion, and substitution.

Globerson and Roweis (2006) present a feature-based learning defense for the *feature deletion* exploratory attack on the evaluation data **E**. In feature deletion, features present in the training data, and perhaps highly predictive of an instance's class, are removed from the evaluation data by the attacker. For example, words present in training emails may not occur in evaluation messages, and network packets in training data may contain values for optional fields that are missing from future traffic. Globerson and Roweis formulate a modified support vector machine classifier that is robust in its choice of features against deletion of high-value features.

One particularly important consideration when the learner builds its model or signature is to ensure that the learner uses features related to the intrusion itself. In their study of the DARPA/Lincoln Laboratory IDS evaluation dataset, Mahoney and Chan (2003) demonstrate that spurious artifacts in training data can cause an IDS to learn to distinguish normal from intrusion traffic based on those artifacts rather than relevant features. Ensuring that the learner builds a model from features that describe the fundamental differences between malicious and benign instances should mitigate the effects of mimicry attacks (Sect. 3.3) and red herring attacks (Sect. 3.1).

Using spurious features in constructing a model or signature is especially problematic in cases where any given intrusion attempt may cause harm only probabilistically or depending on some internal state of the victim's system. If the features relevant to the intrusion are consistent for some set of instances but the actual cost of those instances varies widely, then a learner risks attributing the variation to other nonessential features.

*Hypothesis space/learning procedures* A complex hypothesis space may make it difficult for the attacker to infer precise information about the learned hypothesis. However, hypothesis complexity must be balanced with capacity to generalize, such as through regularization.

Wang et al. (2006) present *Anagram*, an anomaly detection system using *n-gram* models of bytes to detect intrusions. They incorporate two techniques to defeat *Exploratory* attacks that mimic normal traffic (*mimicry attacks*): (1) they use high-order $n$-grams (with $n$ typically between 3 and 7), which capture differences in intrusion traffic even when that traffic has been crafted to mimic normal traffic on the single-byte level; and (2) they randomize feature selection by randomly choosing several (possibly overlapping) subsequences of bytes in the packet and testing them separately, so the attack will fail unless the attacker makes not only the whole packet but also any subsequence mimic normal traffic.

Dalvi et al. (2004) develop a cost-sensitive game-theoretic classification defense to counter *Exploratory Integrity* attacks. In their model, the attacker can alter natural instance features in $A_E$ but incurs a known cost for each change. The defender can measure each feature at a different known cost. Each has a known cost function over classification/true label pairs. The classifier $H$ is a cost-sensitive naive Bayes learner that classifies instances to minimize its expected cost, while the attacker modifies features to minimize its own expected cost. Their defense constructs an adversary-aware classifier by altering the likelihood function of the learner to anticipate the attacker's changes. They adjust the likelihood that an instance is malicious by considering that the observed instance may be the result of an attacker's optimal transformation of another instance. This defense relies on two assumptions: (1) the defender's strategy is a step ahead of the attacker's strategy (in other words,

their game differs from ours in that the attacker's procedure $A_E$ cannot take $f$ into account), and (2) the attacker plays optimally against the original cost-sensitive classifier. It is worth noting that while their approach defends against optimal attacks, it doesn't account for non-optimal attacks. For example, if the attacker doesn't modify any data, the adversary-aware classifier misclassifies some instances that the original classifier correctly classifies.

#### *4.1.2 Defenses against probing attacks*

The ability for $A_E$ to query a classifier gives an attacker powerful additional attack options.

*Analysis of reverse engineering* Lowd and Meek (2005a) observe that the attacker need not model the classifier explicitly, but only find lowest-attacker-cost instances as in the Dalvi et al. setting. They formalize a notion of reverse engineering as the *adversarial classifier reverse engineering* (ACRE) problem. Given an attacker cost function, they analyze the complexity of finding a lowest-attacker-cost instance that the classifier labels as negative. They assume no general knowledge of training data, though the attacker does know the feature space and also must have one positive example and one negative example. A classifier is *ACRE-learnable* if there exists a polynomial-query algorithm that finds a lowest-attacker-cost negative instance. They show that linear classifiers are ACRE-learnable with linear attacker cost functions and some other minor restrictions.

The ACRE-learning problem provides a means of qualifying how difficult it is to use queries to reverse engineer a classifier from a particular hypothesis class using a particular feature space. We now suggest defense techniques that can increase the difficulty of reverse engineering a learner.

*Randomization* A randomized hypothesis may decrease the value of feedback to an attacker. Instead of choosing a hypothesis $f : \mathcal{X} \to \{0, 1\}$, we generalize to hypotheses that predict a real value on $[0, 1]$. This generalized hypothesis returns a probability of classifying $x$ as 1. By randomizing, the expected performance of the hypothesis may decrease on regular data drawn from a non-adversarial distribution, but it also may decrease the value of the queries for the attacker.

Randomization in this fashion does not reduce the information available in principle to the attacker, but merely requires more work from the attacker for the information. It is likely that this defense is appropriate in only a small number of scenarios.

*Limiting/misleading feedback* Another potential defense is to limit the feedback given to an attacker. For example, common techniques in the spam domain include eliminating bounce emails, remote image loading, and other potential feedback channels. It is impossible to remove all feedback channels; however, limiting feedback increases work for the attacker. In some settings, it may be possible to mislead the attacker by sending fraudulent feedback.

Actively misleading the attacker by fabricating feedback suggests an interesting battle of information between attacker and defender. In some scenarios the defender may be able to give the attacker no information via feedback, and in others the defender may even be able to return feedback that causes the attacker to come to incorrect conclusions.

### 4.2 Defending against Causative attacks

In *Causative* attacks, the attacker has a degree of control over not only the evaluation distribution but also the training distribution. Therefore the learning procedures we consider must

be resilient against contaminated training data, as well as to the evaluation considerations discussed in Sect. 4.1.

Two general strategies for defense are to remove malicious data from the training set and to harden the learning algorithm against malicious training data. We first present one method of our own for the former and then describe two approaches to the latter that appear in the literature.

#### *4.2.1 The RONI defense*

We propose the *Reject On Negative Impact* (*RONI*) *defense*, a technique that measures the empirical effect of each training instance and eliminates from training those points that have a substantial negative impact on classification accuracy. To determine whether a candidate training instance is malicious or not, we can train a classifier on a base training set, then add the candidate instance to our training set and train a second classifier. We apply both classifiers to a *quiz set* of instances with known labels, measuring the difference in accuracy between the two. If adding the candidate instance to our training set causes the resulting classifier to produce substantially more classification errors, we reject the instance as detrimental in its effect.

We refine and explore the RONI defense experimentally in Sect. 5.3.

#### *4.2.2 Robustness*

The field of Robust Statistics explores procedures that limit the impact of a small fraction of deviant (adversarial) training data. In the setting of Robust Statistics, it is assumed that the bulk of the data is generated from a known model, but a fraction of the data comes from an unknown model—to bound the effect of this unknown source it is assumed to be adversarial. There are a number of measures of a procedure's robustness: the *breakdown point* is the level of contamination required for the attacker to arbitrarily manipulate the procedure and the *influence function* measures the impact of contamination on the procedure. Robustness measures can be used to assess the susceptibility of an existing system and to suggest alternatives that reduce or eliminate the vulnerability. Ideally one would like to use a procedure with a high breakdown point and a bounded influence function. In this way, these measures can be used to compare candidate procedures and to design procedures $H$ that are optimally robust against adversarial contamination of the training data. For a full treatment, see the books by Huber (1981), Hampel et al. (1986), and Maronna et al. (2006).

Recent research has highlighted the importance of robust procedures in security and learning tasks. Wagner (2004) observes that common sensor net aggregation procedures, such as computing a mean, are not robust to adversarial point contamination, and he identifies robust replacements. Christmann and Steinwart (2004) study robustness for a general family of learning methods. Their results suggest that certain commonly used loss functions, along with proper regularization, lead to robust procedures with a bounded influence function. These results suggest such procedures have desirable properties for secure learning.

#### *4.2.3 Online prediction with experts*

Another approach to combatting *Causative* attacks is to dynamically adapt to the data in an online fashion. In this setting, we allow for the attacker to potentially have arbitrary control of the training data, but instead of attempting to learn on this arbitrarily corrupted data, the

online learner attempts to optimize with respect to a set of *experts*. For instance, the experts may be a set of classifiers each designed to provide different security properties. The experts provide advice (predictions) to the defender who forms a composite classifier that weighs the advice based on each experts past performance and thus produces a composite prediction. We make no assumption about how the experts form their advice or about their performance. However, rather than evaluating the cost of the composite classifier's predictions directly, we instead compare the cost incurred by the composite classifier relative to the cost of the best expert in hindsight; that is, we compute the *regret* the composite classifier has for not heeding the advice of the best expert. In designing strategies that optimize regret, the composite classifiers for these *online* games create a moving target and force the attacker to construct attacks that succeed against a set of experts rather than a single one.

In this setting, the learner forms a prediction from the $M$ expert predictions and adapts the hypothesis based on their performance during $K$ repetitions. At each step $k$ of the game, the defender receives a prediction $g_m^{(k)}$ from each expert; this may be based on the data but we make no assumptions about its behavior. More formally, the $k$-th round of the expert-based prediction game is:

1. **Defender** Update function $h^{(k)} : \mathcal{Y}^M \to \mathcal{Y}$
2. **Attacker** Choose distribution $\mathbb{P}^{(k)}$
3. Evaluation:
   - Sample an instance $(x^{(k)}, y^{(k)}) \sim \mathbb{P}^{(k)}$
   - Compute expert advice $\{g_m^{(k)}\}_{m=1}^M$
   - Predict $\hat{y}^{(k)} = h^{(k)}(g_1^{(k)}, \ldots, g_M^{(k)})$
   - Assess cost $C(x^{(k)}, \hat{y}^{(k)}, y^{(k)})$

This game has a slightly different structure from the games we present in Sect. 2.4—here the defender chooses one strategy at the beginning of the game and then in each iteration updates the function $h^{(k)}$ according to that strategy. The attacker, however, may select a new strategy at each iteration.

The setting of online expert-based prediction splits risk minimization into two subproblems: (1) minimizing the average loss of each expert and (2) minimizing the average *regret*—the difference between the loss of our composite learner and the loss of the best overall expert in hindsight. The other defenses we have discussed approach the first problem. Online game theory addresses the second problem: the defender chooses a strategy for updating $h^{(k)}$ to minimize regret based only on the expert's past performance. For certain variants of the game, there exist composite predictors whose regret is $o(K)$—that is, the average regret approaches 0 as the $K$ increases. Thus, the composite learner can perform almost as well as the best expert without knowing ahead of time which expert is best. A full description of this setting and several results appear in Cesa-Bianchi and Lugosi (2006).

Importantly, the online prediction setting allows the defender to adapt to an adversary and forces the adversary to design attack strategies that succeed against an entire set of experts (each of which can have its own security design considerations). Thus, we can incorporate several classifiers with desirable security properties into a composite approach. Moreover, if a particular attack is succeeding, we can design a new expert against the identified vulnerability and add it to our set of experts to patch the exploit. This makes online prediction well-suited to the changing attack landscape.

## 5 Case study: attacking SpamBayes

We have put our framework to use studying attacks against the SpamBayes statistical spam filter (Nelson et al. 2008). Here we review that work and demonstrate how our framework informs and structures the analysis.

SpamBayes is a content-based statistical spam filter that classifies email using token counts in a model proposed by Robinson (2003) and inspired by Graham (2002). Meyer and Whateley (2004) describe the system in detail. SpamBayes computes a score for each token in the training corpus; this score is similar to a smoothed estimate of the posterior probability that an email containing that token is spam. Each token's score is an estimator of the conditional likelihood that a message is spam given that it contains the token.[2] In terms of (1), the loss function $\ell(\cdot,\cdot)$ takes the form of the logarithm of the posterior probability for a binomial model, the regularization term $\rho(\cdot)$ takes the form of the logarithm of a prior probability using a beta prior to smooth the estimate, and $\lambda = 1$ is the regularizer's weight. The filter computes a message's spam score by assuming token scores are independent and applying Fisher's method for combining significance tests (Fisher 1948). The message score is compared against two thresholds to select the label *spam*, *ham* (non-spam), or *unsure*.

### 5.1 Causative Availability attacks on SpamBayes

In analyzing the vulnerabilities of SpamBayes, we are motivated by our taxonomy of attacks. Known attacks that spammers use against deployed spam filters tend to be *Exploratory Integrity* attacks: either they obfuscate the especially spam-like content of a spam email or they include content not indicative of spam. Both tactics aim to get the modified message into the victim's inbox. This category of attack has been studied in detail in the literature (Lowd and Meek 2005a, 2005b; Wittel and Wu 2004; Dalvi et al. 2004). However, we find the study of *Causative* attacks more compelling because they are unique to machine learning systems and potentially more harmful.

In particular, a *Causative Availability* attack can create a powerful denial of service. For example, if a spammer causes enough legitimate messages to be filtered away by the user's spam filter, the user is likely to disable the filter and therefore see the spammer's advertisements. As another example, an unscrupulous business owner may wish to use spam filter denial of service to prevent a competitor from receiving email orders from potential customers. In this section, we present two novel *Causative Availability* attacks against SpamBayes: the *dictionary* attack is *Indiscriminate* and the *focused* attack is *Targeted*.

We consider an attacker with the following capabilities. We allow the attacker's procedures $A_T$ and $A_E$ to craft email messages with arbitrary message bodies but realistically limited control over the headers. Because we limit our study to *Availability* attacks, we assume that all malicious messages generated by $A_T$ are labeled as *spam* when included in the training set $\mathbf{X}$; if we allow the attacker to create *ham*-labeled training messages then *Integrity* attacks become easier, but the advantage for *Availability* attacks is small.

*Dictionary attack* Our first attack is an *Indiscriminate* attack—the attacker wants to cause a large number of false positives so that the user loses confidence in the filter and must manually sort through spam and ham emails. There are three variants of this attack. In the first, the attacker maximizes the expected spam score of any future message in an *optimal*

[2]The estimator used by Meyer and Whateley (2004) for conditional likelihood deviates slightly from a traditional maximum likelihood estimator.

attack by simply including *all possible tokens* (words, symbols, misspellings, etc.) in attack emails, causing SpamBayes to learn that all tokens are indicative of spam. In practice this optimal attack is intractable, but we approximate its effect by using a large set of common words such as a dictionary—hence these are *dictionary attacks*. The two other variants of the dictionary attack use the *Aspell* dictionary and a dictionary compiled from the most common tokens observed in a *Usenet* corpus.

*Focused attack* Our second attack is a *Targeted* attack—the attacker has some knowledge of a specific legitimate email to target. If the attacker has exact knowledge of the target email, placing all of its tokens in attack emails produces an optimal attack. Realistically, the attacker has partial knowledge about the target email and can guess only some of its tokens to include in attack emails. We model this knowledge by letting the attacker probabilistically guess tokens from the target email. This is the *focused attack*.

## 5.2 Experiments with SpamBayes

We have constructed *Causative Availability* attacks on SpamBayes; here we summarize results of our attack experiments, described in full in our earlier paper (Nelson et al. 2008). We use the Text Retrieval Conference (TREC) 2005 spam corpus (Cormack and Lynam 2005), which is based on the Enron email corpus (Klimt and Yang 2004) and contains 92,189 emails (52,790 spam and 39,399 ham). From this dataset, we construct sample inboxes and measure the effect of injecting our attacks into them.

Figure 1 shows the average effect of our dictionary and focused attacks. In both graphs, the $x$-axis is the contamination percent of the training set. For the dictionary attacks, the $y$-axis is the percent of test ham messages misclassified. For the focused attack, the $y$-axis is the percent misclassification of the *target* message, averaged over 200 random target messages. Although the graphs do not include error bars, we observe that the variation is small.

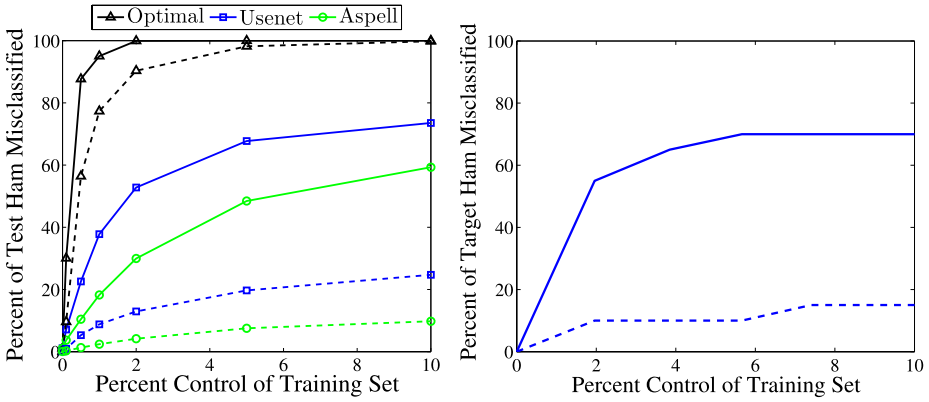


**Fig. 1** Effect of the dictionary and focused attacks. We plot percent of ham classified as *spam* (*dashed lines*) and as *unsure* or *spam* (*solid lines*) against percent of the training set contaminated. *Left*: Three dictionary attacks on an initial training set of 10,000 messages (50% spam). We show the optimal attack (*black* △), the Usenet dictionary attack (*blue* □), and the Aspell dictionary attack (*green* ○). *Right*: The average effect of 200 focused attacks on their targets when the attacker guesses each target token with 50% probability. The initial inbox contains 5000 emails (50% spam)

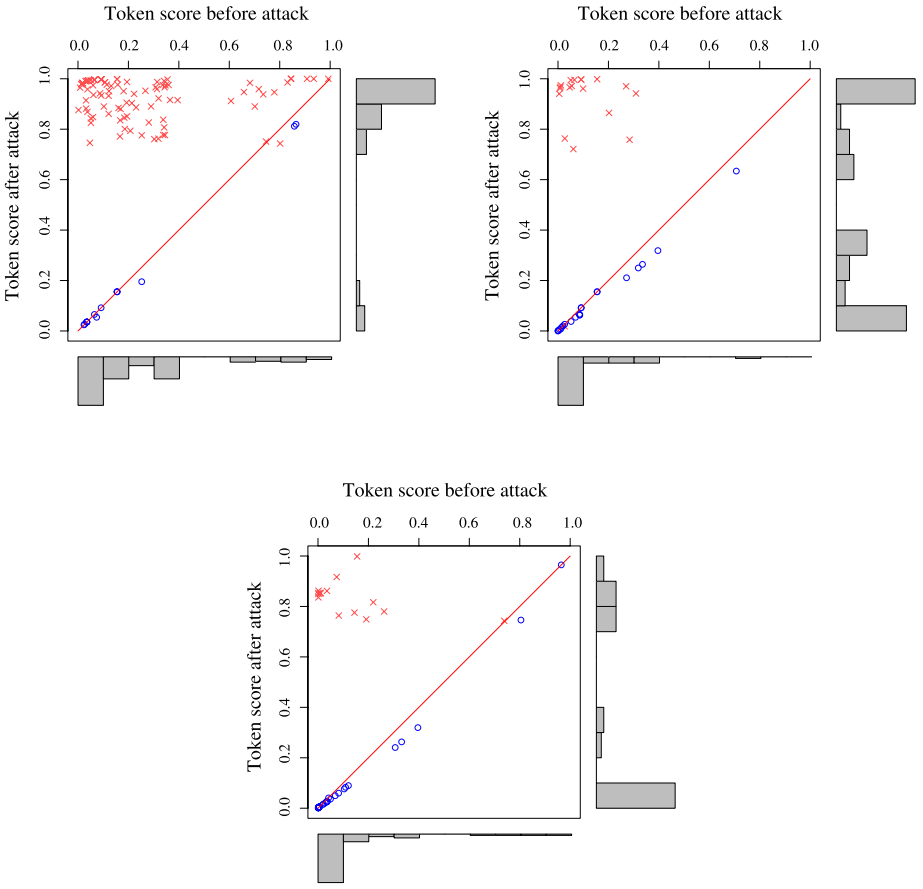


**Fig. 2** Effect of the focused attack on three representative emails. Each point is a token. The $x$-axis is the token spam score before the attack (0 means ham and 1 means spam); the $y$-axis is the spam score after the attack. The *red* ×'s are tokens included in the attack and the *blue* ○'s are tokens that were not in the attack. Histograms show the distribution of token scores before the attack (at *bottom*) and after the attack (at *right*)

The optimal attack quickly causes the filter to mislabel all legitimate emails as spam. The Usenet dictionary attack (90,000 top-ranked words from the Usenet corpus) causes significantly more misclassifications than the Aspell dictionary attack, since it contains common tokens, such as misspellings and slang terms, that are not present in an English dictionary. The focused attack (where each token in the target message is guessed with 50% probability) has an effect on its target comparable to the Usenet dictionary attack.

To better understand the results of the focused attack, we examine its effect on individual tokens. Each of the panels in Fig. 2 represents the tokens from a single target email: the upper-left email is a ham message misclassified as *spam*, the upper-right email is a ham message misclassified as *unsure*, and the bottom-middle email is a ham message correctly classified as *ham*. Each point in the graph represents the before and after spam scores of a token. A point above the line $y = x$ signifies that the token's score increases (becomes more indicative of spam) because of the attack and a point below the line signifies that the token's score decreases. The scores of tokens included in the attack messages typically increase significantly while those not included decrease slightly. The increase in score for included

tokens is more significant than the decrease in score for excluded tokens, so the attack has substantial impact even when the attacker guesses only a fraction of the tokens.

All of our attacks require relatively few attack emails to significantly degrade SpamBayes's accuracy; even the 10% false positive rate induced by the Aspell dictionary attack renders a spam filter unusable.

### 5.3 The Reject On Negative Impact (RONI) defense

In Sect. 4.2.1, we mention the possibility of the Reject On Negative Impact (RONI) defense. As we stated in that section, the RONI defense measures the empirical effect of each training instance and eliminates from training those points that have a substantial negative impact on classification accuracy. To determine whether a candidate training instance is malicious or not, we can train a classifier on a base training set, then add the candidate instance to our training set and train a second classifier. We apply both classifiers to a *quiz set* of instances with known labels, measuring the difference in accuracy between the two. If adding the candidate instance to our training set causes the resulting classifier to produce substantially more classification errors, we reject the instance as detrimental in its effect. In this section we describe the RONI defense in greater detail and discuss our experimental results.

We assume we are given an initial training set $\mathbf{X}$ and a set $\mathbf{Z}$ of additional candidate training points. We evaluate the points in $\mathbf{Z}$ as follows: first we set aside a *calibration set* $\mathbf{C}$, which is a randomly chosen subset of $\mathbf{X}$. Then using the remaining portion of $\mathbf{X}$, we sample a number of independent and potentially overlapping training/quiz set pairs $(\mathbf{T}_i, \mathbf{Q}_i)$, where the points within a pair of sets are sampled without replacement. To assess the impact (empirical effect) of a data point $(x, y) \in \mathbf{Z}$, for each pair of sets $(\mathbf{T}_i, \mathbf{Q}_i)$ we construct a *before* classifier $f_i$ trained on $\mathbf{T}_i$ and an *after* classifier $\hat{f}_i$ trained on $\mathbf{T}_i + (x, y)$. Our RONI defense then compares the classification accuracy of $f_i$ and $\hat{f}_i$ on the quiz set $\mathbf{Q}_i$, using the change in true positives and true negatives caused by adding $(x, y)$ to $\mathbf{T}_i$. If either change is significantly negative when averaged over training/quiz set pairs, we consider $(x, y)$ to be detrimental, so it should not be added to $\mathbf{X}$. To determine the significance of a change, we compare the shift in accuracy to the average shift caused by points in the calibration set $\mathbf{C}$. Each point in $\mathbf{C}$ is evaluated in a way analogous to evaluation of the points in $\mathbf{Z}$. We compute the median and standard deviation of their true positive and true negative changes, and we use the third standard deviation below the median as our significance threshold.

For our experiments with the RONI defense, we again sample inboxes from the TREC 2005 spam corpus. In this assessment, we use 20-fold cross validation to get an initial training inbox $\mathbf{X}$ of about 1000 messages (50% spam) and a test set $\mathbf{E}$ of about 50 messages. We also sample a separate set $\mathbf{Z}$ of 1000 additional messages from the TREC corpus to test as a baseline. In each fold of cross validation, we run five separate trials of the RONI defense. For each trial, we use a calibration set of 25 ham and 25 spam messages, and we sample three training/quiz set pairs of 100 training and 100 quiz messages from the remaining 950 messages. We train two classifiers on each training set for each message in $\mathbf{Z}$, one with and one without the message, measuring performance on the corresponding quiz set and comparing it to the magnitude of change measured from the calibration set.

We perform the RONI defense evaluation for each message in $\mathbf{Z}$ as just described to see the effect on non-attack emails. We find that the RONI defense (incorrectly) rejects an average of 2.8% of the ham and 3.1% of the spam from $\mathbf{Z}$. To evaluate the performance of the post-RONI defense filter, we train a classifier on all messages in $\mathbf{Z}$ and a second classifier on the messages in $\mathbf{Z}$ not rejected by the RONI defense. When trained on all 1000 messages, the resulting filter correctly classifies 97% of ham and 80% of the spam. After removing the

**Table 4** Effect of the RONI defense on the accuracy of SpamBayes in the absence of attacks. Each confusion matrix shows the breakdown of SpamBayes's predicted labels for both ham and spam messages. Left: The average performance of SpamBayes on training inboxes of about 1,000 messages (50% spam). Right: The average performance of SpamBayes after the training inbox is censored using the RONI defense. On average, the RONI defense removes 2.8% of ham and 3.1% of spam from the training sets. (Numbers may not add up to 100% because of rounding error)

Before the RONI defense

| | | Predicted Label | | |
|---|---|---|---|---|
| | | ham | spam | unsure |
| True Label | ham | 97% | 0.0% | 2.5% |
| | spam | 2.6% | 80% | 18% |

After the RONI defense

| | | Predicted Label | | |
|---|---|---|---|---|
| | | ham | spam | unsure |
| True Label | ham | 95% | 0.3% | 4.6% |
| | spam | 2.0% | 87% | 11% |

messages rejected by the RONI defense and training from scratch, the resulting filter still correctly classifies 95% of ham and 87% of the spam. The overall effect of the RONI defense on classification accuracy is shown in Table 4.

Since the RONI defense is removing non-attack emails in this test, and therefore removing potentially useful information from the training data, we expect classification accuracy to be hurt. It is interesting to see that test performance on spam actually improves after removing some emails from the training set. This result seems to indicate that some non-attack emails confuse the filter more than they help when used in training, perhaps because they happen naturally to fit some of the characteristics that attackers use in emails.

Next we evaluate the performance of the RONI defense where $\mathbf{Z}$ instead consists of attack emails from the attacks described earlier in Sect. 5.1. The RONI defense rejects every single dictionary attack from any of the dictionaries (optimal, Aspell, and Usenet). In fact, the degree of change in misclassification rates for each dictionary message is greater than five standard deviations from the median, suggesting that these attacks are easily eliminated with only minor impact on the performance of our filter. See Table 5.

A similar experiment with attack emails from the focused attack shows that the RONI defense is much less effective against focused attack messages. The likely explanation is simple: the dictionary attack broadly affects many different messages with its wide scope of tokens, so its consequences are likely to be seen in our quiz sets. However, the focused attack is targeted at a single *future* email, which may not bear any significant similarity to the messages in the quiz sets. However, as the fraction of tokens correctly guessed by the attacker increases, the RONI defense identifies increasingly many attack messages: only 7% are removed when the attacker guesses 10% of the tokens but 25% of the attacks are removed when the attacker guesses 100% of the tokens. This is likely due to the fact that with more correctly guessed tokens, the overlap with other messages increases sufficiently to trigger the RONI defense more frequently. However, the attack is still successful in spite of the increased number of detections. See Table 6.

The RONI defense is a successful mechanism that thwarts a broad range of dictionary attacks—or more generally *Indiscriminate Causative Availability* attacks. However, the RONI defense also has costs. First, as we show above, this defense yields a slight decrease in ham classification (from 98% to 95%). Second, the RONI defense requires a substantial amount of computation—testing each message in $\mathbf{Z}$ requires us to train and compare the performance of several classifiers. Each new training message must be added to each of the training sets $\mathbf{T}_i$ and the resulting new classifiers must be reevaluated on the quiz sets $\mathbf{Q}_i$. Thus, to add a single message to a classifier, we must first add it to several smaller classifiers (in our case five) then evaluate a set of messages for each (in our case 100 messages

**Table 5** We apply the RONI defense to dictionary attacks with 1% contamination of training inboxes of about 1,000 messages (50% spam) each. Left: The average effect of optimal, Usenet, and Aspell attacks on the SpamBayes filter's classification accuracy. The confusion matrix shows the breakdown of SpamBayes's predicted labels for both ham and spam messages after the filter is contaminated by each dictionary attack. Right: The average effect of the dictionary attacks on their targets after application of the RONI defense. By using the RONI defense, all of these dictionary attacks are caught and removed from the training set, which dramatically improves the accuracy of the filter

| Dictionary Attacks (Before the RONI defense) | | | | |
|---|---|---|---|---|
| | | Predicted Label | | |
| | | ham | spam | unsure |
| Optimal | | | | |
| True Label | ham | 4.6% | 83% | 12% |
| | spam | 0.0% | 100% | 0.0% |
| Aspell | | | | |
| True Label | ham | 66% | 12% | 23% |
| | spam | 0.0% | 98% | 1.6% |
| Usenet | | | | |
| True Label | ham | 47% | 24% | 29% |
| | spam | 0.0% | 99% | 0.9% |

| Dictionary Attacks (After the RONI defense) | | | | |
|---|---|---|---|---|
| | | Predicted Label | | |
| | | ham | spam | unsure |
| Optimal | | | | |
| True Label | ham | 95% | 0.3% | 4.6% |
| | spam | 2.0% | 87% | 11% |
| Aspell | | | | |
| True Label | ham | 95% | 0.3% | 4.6% |
| | spam | 2.0% | 87% | 11% |
| Usenet | | | | |
| True Label | ham | 95% | 0.3% | 4.6% |
| | spam | 2.0% | 87% | 11% |

**Table 6** We apply the RONI defense to focused attacks with 1% contamination of training inboxes of about 1,000 messages (50% spam) each. Left: The average effect of 35 focused attacks on their targets when the attacker correctly guesses 10, 30, 50, 90, and 100% of the target's tokens. Right: The average effect of the focused attacks on their targets after application of the RONI defense. By using the RONI defense, more of the target messages are correctly classified as ham, but the focused attacks still succeed at misclassifying most targeted messages

| Focused Attacks (Before the RONI defense) | | | |
|---|---|---|---|
| | Predicted Label of Target | | |
| | ham | spam | unsure |
| 10% guessed | 78% | 0.0% | 22% |
| 30% guessed | 30% | 5.2% | 65% |
| 50% guessed | 5.8% | 23% | 71% |
| 90% guessed | 0.0% | 79% | 21% |
| 100% guessed | 0.0% | 86% | 14% |

| Focused Attacks (After the RONI defense) | | | |
|---|---|---|---|
| | Predicted Label of Target | | |
| | ham | spam | unsure |
| 10% guessed | 79% | 2.7% | 21% |
| 30% guessed | 36% | 4.8% | 59% |
| 50% guessed | 19% | 20% | 61% |
| 90% guessed | 20% | 62% | 19% |
| 100% guessed | 21% | 66% | 13% |

per classifier). While this RONI defense method can be parallelized, this additional work does increase the training time required; our non-parallelized, non-optimized experiments increased training time by two orders of magnitude because we trained a full classifier several times for each message in the training set. Finally, the RONI defense may slow the learning process. For instance, when a user correctly labels a new type of spam for training, the RONI defense may reject those instances because the new spam may be very different from spam previously seen and more similar to some non-spam messages in the training set. This effect is hard to quantify and may or may not be an issue in practice.

We believe there is still ample room for optimization and refinement of the RONI defense that may ameliorate one or both of these problems. The question of whether the RONI

defense is cost effective in practice is an open one, depending in part on the resource limitations and attack risks of any particular environment.

## 6 Discussion

Our framework opens a number of new research directions. One of the most promising research directions is measuring the amount of information leaked from a learning system to an attacker. An adversary may try to gain information about the internal state of a machine learning system to: (a) extract personal information encoded in the internal state (for example, in a machine learning anti-spam system, knowledge about how certain keywords are handled may leak information about the contents or senders of typical email messages handled by that system); or (b) derive information that will allow the adversary to more effectively attack the system in the future.

We can measure the amount of information leaked in terms of the number of bits of the information, but note that different pieces of information may be more or less relevant to the security of a system than others. For example, exposing a person's 9-digit social security number can be very damaging, but exposing that person's 9-digit postal code is much more benign. This suggests an open problem: is it possible to develop a "theory of information for security" that can measure the value of leaked information? An answer to this question is likely to build on Shannon's classical theory of information (Shannon 1948) as well as computationally-based variants of it due to Kolmogorov (1993) and Yao (1988). A "theory of information for security" could have wide applicability, not only in the context of understanding adversarial attacks on machine learning system, but also in quantifying the risk associated with various *side channel attacks* that exploit leaked information.

We also open new directions for evaluating defenses. In Sect. 4, we introduce several promising ideas for defenses against learning attacks. The next step is to explore general defenses against larger classes of attack.

Developing a general framework for constructing and evaluating defenses would be a valuable contribution. Measuring the adversarial effort required to perform an attack as well as the effectiveness of the defense could help design secure learning systems. The ACRE-learning framework of Lowd and Meek (2005a) provides a computational analysis of the complexity of reverse engineering a hypothesis using queries in an *Exploratory* attack. The problem of *Causative* attacks may be more difficult; here the fields of robust statistics and online prediction games provide a foundation on which to build new defenses.

An interesting aspect of attacks on machine learning is the link between complexity and attacks. In some cases, greater model complexity seems to confer advantage: text generated by a bigram model cannot be distinguished from its source material by a unigram model, but a trigram model can differentiate them. Wang et al. (2006) demonstrate that an increase in model complexity can defend against some attacks. Similarly, Tan and Maxion (2002) have shown that a sequence-based IDS must use a sequence length at least as large as the *minimal foreign subsequence* of an attack, or smallest necessary subsequence of events in the attack, for the detector to be successful (see also earlier work on mimicry attacks in Sect. 3.3). Does this indicate a general trend? Can the advantages gained from a more complex model offset the chance of overfitting, the need for more training data, and other downsides of model complexity?

## 7 Conclusion

We have presented a framework for articulating a comprehensive view of different classes of attacks on machine learning systems in terms of three independent dimensions and an adversarial learning game. Guided by our framework, we survey relevant prior research and explore the effects of different types of learning attacks on systems and their defenses against these attacks. We provide a concrete extended example by applying our framework to a machine learning-based application, the SpamBayes spam detection system, and show how the attacks motivated by our framework can successfully cause SpamBayes to fail. We also develop and demonstrate a concrete defense that succeeds against *Indiscriminate Causative Availability* attacks on SpamBayes and has potential for wider applicability. Our framework provides a solid foundation for analyzing attacks on machine learning systems and developing defenses against them, taking a significant step towards the goal of secure learning.

**Acknowledgements** We would like to thank Russell Sears, Benjamin Rubinstein, David Molnar, Peter Bartlett, Michael Jordan, Satish Rao, Carla Brodley, and our anonymous reviewers for their insightful discussions, comments and suggestions regarding this research.

We gratefully acknowledge the support of our sponsors. This work was supported in part by TRUST (Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation (NSF award number CCF-0424422) and the following organizations: AFOSR (#FA9550-06-1-0244), BT, Cisco, DoCoMo USA Labs, EADS, ESCHER, HP,IBM, iCAST, Intel, Microsoft, ORNL, Pirelli, Qualcomm, Sun, Symantec,TCS, Telecom Italia, and United Technologies; in part by RAD Lab (Reliable Adaptive Distributed Systems Laboratory), which receives support from California state Microelectronics Innovation and Computer Research Opportunities grants (MICRO ID#06-148 and #07-012) and the following organizations: Amazon Web Services, CISCO, Cloudera, eBay, Facebook, Fujitsu Labs of America, Google, Hewlett Packard, Intel, Microsoft, NetApp, SAP, Sun, VMWare, and Yahoo!; 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 here are solely those of the authors and do not necessarily reflect the opinions of any funding agency, the State of California, or the US government.



## References

Barreno, M., Nelson, B., Sears, R., Joseph, A. D., & Tygar, J. D. (2006). Can machine learning be secure? In *Proceedings of the ACM symposium on InformAtion, computer, and communications security (ASIACCS)*.

Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge: Cambridge University Press.

Christmann, A., & Steinwart, I. (2004). On robustness properties of convex risk minimization methods for pattern recognition. *Journal of Machine Learning Research (JMLR)*, *5*, 1007–1034.

Chung, S. P., & Mok, A. K. (2006). Allergy attack against automatic signature generation. In *Recent advances in intrusion detection (RAID)* (pp. 61–80).

Chung, S. P., & Mok, A. K. (2007). Advanced allergy attacks: Does a corpus really help? In *Recent advances in intrusion detection (RAID)* (pp. 236–255).

Cormack, G., & Lynam, T. (2005). Spam corpus creation for TREC. In *Proceedings of the conference on email and anti-spam (CEAS)*.

Dalvi, N., Domingos, P., Mausam, Sanghai, S., & Verma, D. (2004). Adversarial classification. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 99–108).
Dredze, M., Gevaryahu, R., & Elias-Bachrach, A. (2007). Learning fast classifiers for image spam. In *Proceedings of the conference on email and anti-spam (CEAS)*.
Fisher, R. A. (1948). Question 14: Combining independent tests of significance. *American Statistician*, *2*(5), 30–31.
Fogla, P., & Lee, W. (2006). Evading network anomaly detection systems: Formal reasoning and practical techniques. In *Proceedings of the ACM conference on computer and communications security (CCS)* (pp. 59–68).
Globerson, A., & Roweis, S. (2006). Nightmare at test time: Robust learning by feature deletion. In *Proceedings of the international conference on machine learning (ICML)* (pp. 353–360).
Graham, P. (2002). A plan for spam. http://www.paulgraham.com/spam.html.
Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., & Stahel, W. A. (1986). *Robust statistics: the approach based on influence functions. Probability and mathematical statistics*. New York: Wiley.
Huber, P. J. (1981). *Robust statistics. Probability and mathematical statistics*. New York: Wiley.
Kearns, M., & Li, M. (1993). Learning in the presence of malicious errors. *SIAM Journal on Computing*, *22*, 807–837.
Kim, H. A., & Karp, B. (2004). Autograph: Toward automated, distributed worm signature detection. In *USENIX security symposium*.
Klimt, B., & Yang, Y. (2004). Introducing the Enron corpus. In *Proceedings of the conference on email and anti-spam (CEAS)*.
Kolmogorov, A. N. (1993). *Information theory and the theory of algorithms, selected works of A.N. Kolmogorov, Vol. III*. Dordrecht: Kluwer.
Lowd, D., & Meek, C. (2005a). Adversarial learning. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 641–647).
Lowd, D., & Meek, C. (2005b). Good word attacks on statistical spam filters. In *Proceedings of the conference on email and anti-spam (CEAS)*.
Mahoney, M., & Chan, P. (2003). *An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. Lecture notes in computer science* (pp. 220–238).
Maronna, R. A., Martin, D. R., & Yohai, V. J. (2006). *Robust statistics: theory and methods*. New York: Wiley.
Meyer, T. A., & Whateley, B. (2004). SpamBayes: Effective open-source, Bayesian based, email classification system. In *Proceedings of the conference on email and anti-spam (CEAS)*.
Moore, D., Shannon, C., Brown, D. J., Voelker, G. M., & Savage, S. (2006). Inferring internet denial-of-service activity. *ACM Transactions on Computer Systems*, *24*(2), 115–139.
Nelson, B., Barreno, M., Chi, F. J., Joseph, A. D., Rubinstein, B. I. P., Saini, U., Sutton, C., Tygar, J. D., & Xia, K. (2008). Exploiting machine learning to subvert your spam filter. In *Proceedings of the workshop on large-scale exploits and emerging threats (LEET)*.
Newsome, J., Karp, B., & Song, D. (2005). Polygraph: Automatically generating signatures for polymorphic worms. In *Proceedings of the IEEE symposium on security and privacy* (pp. 226–241).
Newsome, J., Karp, B., & Song, D. (2006). Paragraph: Thwarting signature learning by training maliciously. In *Recent advances in intrusion detection (RAID)*.
Robinson, G. (2003). A statistical approach to the spam problem. *Linux Journal*.
Sculley, D., Wachman, G. M., & Brodley, C. E. (2006). Spam filtering using inexact string matching in explicit feature space with on-line linear classifiers. In *Proceedings of the text retrieval conference (TREC)*.
Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*, 379–423 and 623–656.
Tan, K. M. C., & Maxion, R. A. (2002). "Why 6?" Defining the operational limits of `stide`, an anomaly-based intrusion detector. In *Proceedings of the IEEE symposium on security and privacy* (pp. 188–201).
Tan, K. M. C., Killourhy, K. S., & Maxion, R. A. (2002). Undermining an anomaly-based intrusion detection system using common exploits. In *Recent advances in intrusion detection (RAID)* (pp. 54–73).
Tan, K. M. C., McHugh, J., & Killourhy, K. S. (2003). Hiding intrusions: From the abnormal to the normal and beyond. In *Revised papers from the international workshop on information hiding (IH)* (pp. 1–17).
Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, *27*(11), 1134–1142.
Valiant, L. G. (1985). Learning disjunctions of conjunctions. In *Proceedings of the international joint conference on artificial intelligence (IJCAI)* (pp. 560–566).
Wagner, D. (2004). Resilient aggregation in sensor networks. In *Proceedings of the ACM workshop on security of Ad Hoc and sensor networks (SASN)* (pp. 78–87).
Wagner, D., & Soto, P. (2002). Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the ACM conference on computer and communications security (CCS)* (pp. 255–264).

Wang, K., Parekh, J. J., & Stolfo, S. J. (2006). Anagram: A content anomaly detector resistant to mimicry attack. In *Recent advances in intrusion detection (RAID)* (pp. 226–248).

Wang, Z., Josephson, W., Lv, Q., Charikar, M., & Li, K. (2007). Filtering image spam with near-duplicate detection. In *Proceedings of the conference on email and anti-spam (CEAS)*.

Wittel, G. L., & Wu, S. F. (2004). On attacking statistical spam filters. In *Proceedings of the conference on email and anti-spam (CEAS)*.

Yao, A. C. (1988). Computational information theory. In Y. Abu-Mostafa (Ed.), *Complexity in information theory* (pp. 1–15). Berlin: Springer.