# The Security of Machine Learning

*Marco Barreno*
*Blaine Alan Nelson*
*Anthony D. Joseph*
*Doug Tygar*

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2008-43
http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-43.html

April 24, 2008

Acknowledgement

This work was supported in part by the Team for Research in Ubiquitous Secure Technology (TRUST), which receives support from the National Science Foundation (NSF award #CCF-0424422), the Air Force Office of Scientific Research (AFOSR #FA9550-06-1-0244), Cisco, British Telecom, ESCHER, Hewlett-Packard, IBM, iCAST, Intel, Microsoft, ORNL, Pirelli, Qualcomm, Sun, Symantec, Telecom Italia, and United Technologies; in part by California state Microelectronics Innovation and Computer Research Opportunities grants (MICRO ID #06-148, #07-012) and Siemens; and in part by the cyber-DEfense Technology Experimental Research laboratory (DETERlab), which receives support from the Department of Homeland Security's Advanced Research Projects Agency

(HSARPA award #022412) and AFOSR (#FA9550-07-1-0501).

# The Security of Machine Learning

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

University of California, Berkeley
{barreno,nelsonb,adj,tygar}@cs.berkeley.edu

April 2008

**Abstract**

Machine learning has become a fundamental tool for computer security since it can rapidly evolve to changing and complex situations. 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.

## 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.* The question we face is: how do we evaluate the quality of a learning system and determine whether it satisfies requirements for secure learning?

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 (Section 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 their influence over 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 (Section 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 (Section 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 (Section 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.

Our 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. This is 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 and setup

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.

In the *supervised classification* problem, the learner trains on a dataset of $N$ instances, $\mathbf{X} = \{(x,y) \mid x \in \mathscr{X}, y \in \mathscr{Y}\}^N$, given an instance space $\mathscr{X}$ and the label space $\mathscr{Y} = \{0,1\}$. 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 clas-

| | |
|---|---|
| $\mathscr{X}$ | Space of data instances |
| $\mathscr{Y}$ | Space of data labels; for classification $\mathscr{Y} = \{0,1\}$ |
| $\mathfrak{D}$ | Space of distributions over $(\mathscr{X} \times \mathscr{Y})$ |
| $\Omega$ | Space of hypotheses $f : \mathscr{X} \mapsto \mathscr{Y}$ |
| $\mathbb{P}_{\mathrm{T}} \in \mathfrak{D}$ | Training distribution |
| $\mathbb{P}_{\mathrm{E}} \in \mathfrak{D}$ | Evaluation distribution |
| $\mathbb{P} \in \mathfrak{D}$ | Distribution for training and evaluation (Section 4.2.2) |
| $x \in \mathscr{X}$ | Data instance |
| $y \in \mathscr{Y}$ | Data label |
| $\mathbf{X}, \mathbf{E} \in (\mathscr{X} \times \mathscr{Y})^N$ | Datasets |
| $H : (\mathscr{X} \times \mathscr{Y})^N \mapsto \Omega$ | Procedure for selecting hypothesis |
| $A_T, A_E : \mathscr{X}^N \times \Omega \mapsto \mathfrak{D}$ | Procedures for selecting distribution |
| $\ell : \mathscr{Y} \times \mathscr{Y} \mapsto \mathbb{R}^{0+}$ | Loss function |
| $C : \mathscr{X} \times \mathscr{Y} \times \mathscr{Y} \mapsto \mathbb{R}$ | Cost function |
| $f : \mathscr{X} \mapsto \mathscr{Y}$ | Hypothesis (classifier) |
| $f^* : \mathscr{X} \mapsto \mathscr{Y}$ | Best hypothesis |
| $N$ | Number of data points |
| $K$ | Number of repetitions of a game |
| $M$ | Number of experts (Section 4.2.2) |

Table 1: Notation in this paper.

sifier 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.

The procedure can be any method of selecting a hypothesis; in statistical machine learning, the most common type of 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. 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 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.

We assume the use of a classifier for a security goal. For example, a virus detection system has the goal of preventing virus infection, and an intrusion detection system has the goal of preventing malicious intrusion. A virus infection or a successful intrusion has high cost relative to other outcomes. 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 (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 aligned. An important special case is a *zero-sum game*, in which the sum of the attacker's cost and the defender's cost is zero (or any other fixed value) for each possible outcome. In this paper, we assume that games are zero-sum. 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. 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.

In general we assume the attacker can generate arbitrary instances; however, many specific problems impose reasonable restrictions on the attacker's ability to generate instances. 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. 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 Section 4.2.2). 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). In one sense, *Causative* attacks are more fundamentally learning attacks than *Exploratory* attacks are: while many types of systems perform poorly on cleverly modified instances, only systems that learn from data can be misled by an attacker to form incorrect models, choosing poor hypotheses. On the other hand, the hypotheses produced by learning algorithms have certain regularities and structures that an attacker may be able to exploit in an *Exploratory* attack, so it is certainly worthwhile to consider them carefully alongside *Causative* attacks.

The second axis indicates the type of security violation the attacker causes: (a) to create false negatives, in which harmful instances slip through the filter (an *Integrity* violation); or (b) to create a denial of service, usually by inducing false positives, in which benign instances are incorrectly filtered (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)

| | *Integrity* | *Availability* |
|---|---|---|
| *Causative:* | | |
| *Targeted* | *The intrusion foretold*: mis-train a particular intrusion | *The rogue IDS*: mis-train IDS to block certain traffic |
| *Indiscriminate* | *The intrusion foretold*: mis-train any of several intrusions | *The rogue IDS*: mis-train IDS to broadly block traffic |
| *Exploratory:* | | |
| *Targeted* | *The shifty intruder*: obfuscate a chosen intrusion | *The mistaken identity*: censor a particular host |
| *Indiscriminate* | *The shifty intruder*: obfuscate any intrusion | *The mistaken identity*: interfere with traffic generally |

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

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 Section 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 an intrusion detection system (IDS) that uses machine learning. Table 2 summarizes the taxonomy and shows where these examples fit within it. This section gives the reader an intuition for how the taxonomy organizes attacks against machine learning systems; Section 3 presents a similar table categorizing attacks published in the literature.

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

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

Example: an attacker wants the defender's IDS not to block a novel virus. The defender trains periodically on network traffic, so the attacker sends non-intrusion traffic that is carefully chosen to look like the virus and mis-train the learner to fail to block it.

This example might be *Targeted* if the attacker already has a particular virus executable to send and needs to cause the learner to miss that particular instance. It might be *Indiscriminate*, on the other hand, if the attacker has a certain payload but could use any of a large

number of existing exploit mechanisms to transmit the payload, in which case the attack need only fool the learner on any one of the usable executables.

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

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

Example: an attacker wants traffic to be blocked so the destination doesn't receive it. The attacker generates attack traffic similar to benign traffic when the defender is collecting training data to train the IDS. When the learner re-trains on the attack data, the IDS will start to filter away benign instances as if they were intrusions.

This attack could be *Targeted* at a particular protocol or destination. On the other hand, it might be *Indiscriminate* and attempt to block a significant portion of all legitimate traffic.

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

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

Example: an attacker modifies and obfuscates intrusions, such as by changing network headers and reordering or encrypting contents. If successful, these modifications prevent the IDS from recognizing the altered intrusions as malicious, so it allows them into the system.

In the *Targeted* version of this attack, the attacker has a particular intrusion to get past the filter. In the *Indiscriminate* version, the attacker has no particular preference and can search for any intrusion that succeeds, such as by modifying a large number of different exploits to see which modifications evade the filter.

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

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

Example: an attacker sends intrusions that appear to come from the IP address of a legitimate machine. The IDS, which has learned to recognize intrusions, blocks that machine.

In the *Targeted* version, the attacker has a particular machine to target. In the *Indiscriminate* version, the attacker may select any convenient machine or may switch IP addresses among many machines to induce greater disruption.

## 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).

### 2.4.1 Exploratory game

We first present the game for *Exploratory* attacks:

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. 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 is 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 may be based 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 $\mathbf{E}$.

In many cases, the procedure $A_E$ can *query* the classifier, treating it as an oracle that provides labels for query instances. 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.

| | *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 [2005a], Wittel and Wu [2004], Lowd and Meek [2005b] | Moore et al. [2006] |
| *Indiscriminate* | Fogla and Lee [2006], Lowd and Meek [2005a], Wittel and Wu [2004] | Moore et al. [2006] |

Table 3: Related work in the taxonomy.

Control over data used for training opens up new strategies to the attacker. Cost is based on the interaction of $f$ and **E**. In the *Exploratory* game the attacker chooses **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 examples. 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, Karp, and Song [2006] present *Causative Integrity* and *Causative Availability* attacks against Polygraph [Newsome, Karp, and Song, 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*).

### 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).

**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.

**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 Section 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 `stide`.** Tan, Killourhy, and Maxion [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.

**Good word attacks.** Several authors demonstrate *Exploratory integrity* attacks using similar principles against spam filters. Lowd and Meek [2005a] 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 [2005b] 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 Section 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, Shannon, Brown, Voelker, and Savage [2006].

However, *Exploratory Availability* attacks against the learning components of systems are not common. We describe one possibility in Section 2.3.4: if a learning IDS 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 IDS 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.

## 4 Defenses: Applying Our Framework

We discuss several defense strategies against broad classes of attacks. The game between attacker and defender and the taxonomy that we introduce in Section 3 provides a foundation on which to construct defenses. We address *Exploratory* and *Causative* attacks separately, and we also discuss the broader setting of an iterated game. In all cases, we must expect a trade-off: changing the algorithms to make them more robust against (worst-case) attacks will generally make them *less* effective on average. Analyzing and addressing 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. The attacker attempts to construct an *unfavorable evaluation distribution* concentrating probability mass on high-cost instances; in other words, the attacker tries to find an evaluation distribution on which the learner predicts poorly (violating stationarity). 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, the attacker 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 it.

**Feature selection.** We can make classifiers hard to reverse engineer through feature selection. Feature selection is the process of choosing a feature map that maps raw measurements into a new feature space on which a hypothesis is selected. Keeping secret which features are selected in learning, or choosing a secret mapping to a different feature space entirely, may hinder an attacker in finding high-cost instances.

Globerson and Roweis [2006] present a defense for the *Exploratory* attack of *feature deletion* on the evaluation data: 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 robust against deletion of high-value features.

Obfuscation of spam-indicating words (an attack on the feature set) is a common *Targeted Exploratory Integrity* attack. Sculley, Wachman, and Brodley [2006] use inexact string matching to defeat obfuscations of words in spam emails. The use features based on character subsequences that are robust to character addition, deletion, and substitution.

**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, Parekh, and Stolfo [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, Domingos, Mausam, Sanghai, and Verma [2004] develop a cost-sensitive game-theoretic classification defense to counter *Exploratory Integrity* attacks. In their model, the attacker can alter instance features 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 (not zero-sum). The classifier 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, 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 to query a classifier gives an attacker powerful additional attack options.

**Analysis of reverse engineering.** Lowd and Meek [2005b] 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 $f : \mathscr{X} \to \{0,1\}$, 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 the previous section. We consider two different approaches.

### 4.2.1 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 small fraction of the data is selected adversarially. A number of tools exist for assessing robustness: *qualitative robustness*, the *breakdown point* of a procedure (how much data the attacker needs for arbitrary control), and the *influence function* of a procedure (to measure the impact of contamination on the procedure). These tools can be used to design procedures that are 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 learning methods that can be expressed as regularized convex risk minimization on a Hilbert space. Their results suggest that certain commonly used loss functions, along with regularization, lead to robust procedures in the sense of bounded influence. These results suggest such procedures have desirable properties for secure learning.

### 4.2.2 Online prediction with experts

When the attacker has arbitrary control of the training data, the situation is more dire for the defender. If $f$ minimizes risk on the training set, the attacker could choose $A_T$ and $A_E$ to make the evaluation risk approach its maximum. However, with a slight change to the defender's objective, a variant of the iterated *Causative* game yields interesting results.

Consider the case where the attacker has complete control over training data but the defender receives the advice of *M experts* who provide predictions. For example, the defender may have $M$ different classifiers, each of which is designed to be robust in a different way. Each classifier is an expert in this model. We construct a composite classifier that predicts based on the *advice* of the experts. We make no assumptions about how the experts perform, but we evaluate our learner's performance relative to the best expert in hindsight. The

intuition is that the attacker must design attacks that are successful not only against a single expert, but uniformly against the set of experts. The composite learner can perform almost as well as the best one without knowing ahead of time which expert is best.

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 $h^{(k)} : \mathscr{Y}^M \to \mathscr{Y}$
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 $\hat{y}^{(k)} = h^{(k)}\left(g_1^{(k)}, \ldots, g_M^{(k)}\right)$.
   - Assess cost $C\left(x^{(k)}, \hat{y}^{(k)}, y^{(k)}\right)$

This game has a slightly different structure from the games we present in Section 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 allows us to split 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 experts'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. A full description of this setting and several results appear in Cesa-Bianchi and Lugosi [2006].

# 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 resembles a smoothed estimate of the posterior probability that an email containing that token is spam. It 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

We present two *Causative Availability* attacks against SpamBayes.

**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* 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 attack, 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.

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.

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.

Each of the panels in Figure 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

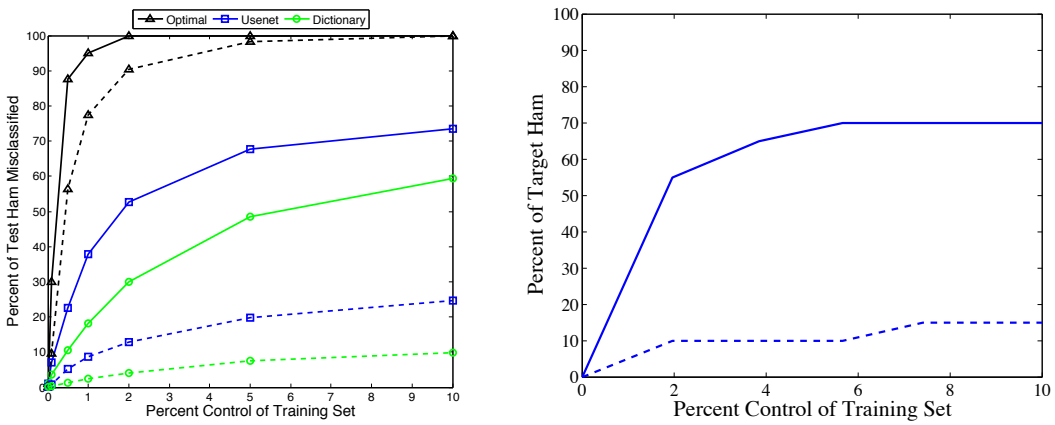


Figure 1: Effect of the dictionary and focused attacks. We plot percent of ham classified as *spam* (dashed lines) and as *unsure* or *spam* (solid lines) against percent of the training set contaminated. **Left:** Three dictionary attacks on an initial training set of 10,000 messages (50% spam). We 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).

classified as *ham*. Each point in the graph represents the before/after score of a token; a point above the line $y = x$ increases (is more indicative of spam) due to the attack and any point below the line 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.

### 5.3 Defenses

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 negatively affect classification. To measure the effect of a query email, we train two learners with identical training sets except that only one includes the query email. If that model performs significantly worse on a test set than the one without, we exclude the query email from our training set.

Preliminary experiments show that the RONI defense is extremely successful against Aspell dictionary attacks, able to identify 100% of the attack emails without a single mistake on non-attack emails. However, the RONI defense fails to differentiate focused attack emails from non-attack emails. The explanation is simple: the dictionary attack broadly affects emails, including training emails, while the focused attack is targeted at a *future* email, so its effects may not be evident on the training set alone.

The RONI defense shows potential to successfully identify a broad range of attacks with further refinement.

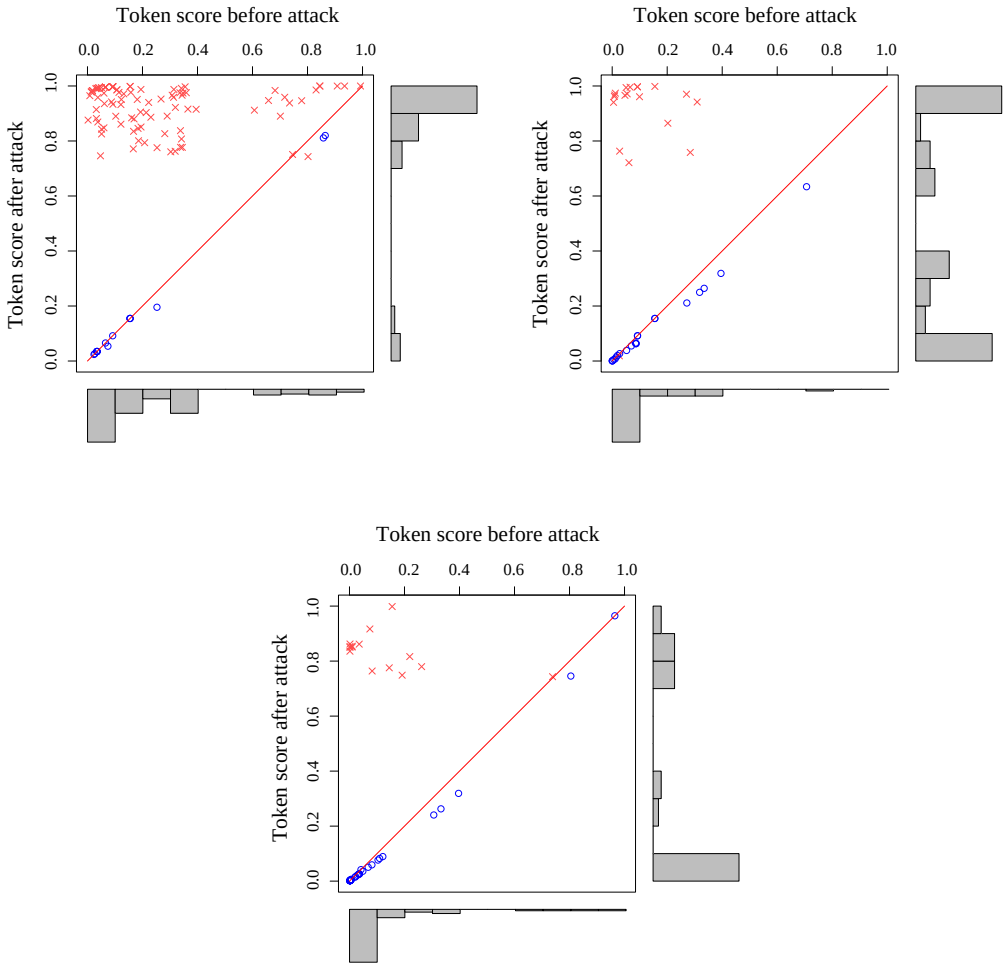


Figure 2: Effect of the focused attack on three representative emails. Each point is a token. The $x$-axis is the token spam score before the attack (0 means ham and 1 means spam); the $y$-axis is the spam score after the attack. The red $\times$'s are tokens included in the attack and the blue $\bigcirc$'s 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).

# 6 Discussion

## 6.1 A theory of information for security

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 bits carry different amounts of useful information. As an extreme example, if the learning system leaks a random bit, the adversary gains no useful information. 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.

## 6.2 Evaluating defenses

In Section 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 [2005b] 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.

## 6.3 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. 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 de-

veloped the notion of a secure learning game. Guided by our framework and the learning game, we identify where relevant prior research fits into the framework. Specifically, we explore the effects of different types of attacks on the systems and their defenses against these attacks.

We have provided a concrete example by applying our framework to a machine learning-based application, the SpamBayes spam detection system, and show how the attacks described by our framework can successfully cause SpamBayes to fail. We demonstrate a concrete defense that reduces the effects of the dictionary attack on SpamBayes.

We believe that our framework opens a number of new research directions. In particular, from the framework, we can generalize to the idea that many of the classes of attacks are dependent upon knowledge that the attacker has gained about the internal states of the learner. Thus, one potentially interesting avenue for future exploration is the idea of securing learning systems by measuring and bounding the amount of information leaked from a learning system to an attacker.

## Acknowledgments

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

## References

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

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

Andreas Christmann and Ingo Steinwart. On robustness properties of convex risk minimization methods for pattern recognition. *Journal of Machine Learning Research (JMLR)*, 5: 1007–1034, 2004. ISSN 1533-7928.

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

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

Gordon Cormack and Thomas Lynam. Spam corpus creation for TREC. In *Proceedings of the 2nd Conference on Email and Anti-Spam (CEAS 2005)*, July 2005.

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

Mark Dredze, Reuven Gevaryahu, and Ari Elias-Bachrach. Learning fast classifiers for image spam. In *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, 2007.

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

Prahlad Fogla and Wenke Lee. Evading network anomaly detection systems: Formal reasoning and practical techniques. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 59–68, 2006.

Amir Globerson and Sam Roweis. Nightmare at test time: Robust learning by feature deletion. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 353–360, 2006.

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

Frank R. Hampel, Elvezio M. Ronchetti, Peter J. Rousseeuw, and Werner A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. Probability and Mathematical Statistics. John Wiley and Sons, 1986.

Peter J. Huber. *Robust Statistics*. John Wiley and Sons, 1981.

Michael Kearns and Ming Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22:807–837, 1993. URL `citeseer.ist.psu.edu/kearns93learning.html`.

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

Bryan Klimt and Yiming Yang. Introducing the Enron corpus. In *Proceedings of the 1st Conference on Email and Anti-Spam (CEAS 2004)*, July 2004.

A. N. Kolmogorov. *Information Theory and the Theory of Algorithms*, volume III of *Selected Works of A.N. Kolmogorov*. Kluwer, 1993.

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

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

Ricardo A. Maronna, Douglas R. Martin, and Victor J. Yohai. *Robust Statistics: Theory and Methods*. John Wiley and Sons, New York, 2006.

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

David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, and Stefan Savage. Inferring internet denial-of-service activity. *ACM Transactions on Computer Systems*, 24(2):115–139, 2006. ISSN 0734-2071.

Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, Udam Saini, Charles Sutton, J. D. Tygar, and Kai Xia. Exploiting machine learning to subvert your spam filter. In *Proceedings of the First Workshop on Large-scale Exploits and Emerging Threats (LEET)*, 2008.

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

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

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

D. Sculley, Gabriel M. Wachman, and Carla E. Brodley. Spam filtering using inexact string matching in explicit feature space with on-line linear classifiers. In *Proceedings of the Fifteenth Text Retrieval Conference (TREC)*, 2006.

C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.

Kymie M. C. Tan, Kevin S. Killourhy, and Roy A. Maxion. Undermining an anomaly-based intrusion detection system using common exploits. In *Recent Advances in Intrusion Detection (RAID)*, pages 54–73, 2002.

Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

Leslie G. Valiant. Learning disjunctions of conjunctions. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 560–566, 1985.

David Wagner. Resilient aggregation in sensor networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN '04)*, pages 78–87, New York, NY, USA, 2004. ACM Press.

Ke Wang, Janak J. Parekh, and Salvatore J. Stolfo. Anagram: A content anomaly detector resistant to mimicry attack. In *Recent Advances in Intrusion Detection (RAID)*, pages 226–248, 2006.

Zhe Wang, William Josephson, Qin Lv, Moses Charikar, and Kai Li. Filtering image spam with near-duplicate detection. In *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, 2007.

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

A. C. Yao. Computational information theory. In Y. Abu-Mostafa, editor, *Complexity in Information Theory*, pages 1–15. Springer-Verlag, 1988.