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 Ω, 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* ℓ that approximates true cost; the regularization term ρ captures some notion of hypothesis complexity to prevent *overfitting* the training data, using a weight λ to quantify the trade-off. This procedure finds the hypothesis minimizing:

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

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 $(\mathcal{X} \times \mathcal{Y})$
Ω	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)
λ	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.