Adversarial Machine Learning[*]

Ling Huang
Intel Labs Berkeley
ling.huang@intel.com

Anthony D. Joseph
UC Berkeley
adj@cs.berkeley.edu

Blaine Nelson
University of Tübingen
blaine.nelson@wsii.uni-tuebingen.de

Benjamin I. P. Rubinstein
Microsoft Research
ben.rubinstein@microsoft.com

J. D. Tygar
UC Berkeley
tygar@cs.berkeley.edu

ABSTRACT

In this paper (expanded from an invited talk at AISEC 2010), we discuss an emerging field of study: adversarial machine learning—the study of effective machine learning techniques against an adversarial opponent. In this paper, we: give a taxonomy for classifying attacks against online machine learning algorithms; discuss application-specific factors that limit an adversary's capabilities; introduce two models for modeling an adversary's capabilities; explore the limits of an adversary's knowledge about the algorithm, feature space, training, and input data; explore vulnerabilities in machine learning algorithms; discuss countermeasures against attacks; introduce the evasion challenge; and discuss privacy-preserving learning techniques.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Invasive software (e.g., viruses, worms, Trojan horses); I.5.1 [**Models**]: Statistical; I.5.2 [**Design Methodology**]

Keywords

Adversarial Learning, Computer Security, Game Theory, Intrusion Detection, Machine Learning, Security Metrics, Spam Filters, Statistical Learning

1. INTRODUCTION

In this paper, we discuss an emerging field of study: adversarial machine learning—the study of effective machine learning techniques against an adversarial opponent. To see why this field is needed, is learning—the study of effective machine learning techniques against an adversarial opponent. To see why this field is needed, it is helpful to recall a common metaphor: security is sometimes thought of as a chess game between two players. For a player to win, it is not only necessary to have an effective strategy, one must also anticipate the opponent's response to that strategy.

Statistical machine learning has already become an important tool in a security engineer's repertoire. However, machine learning in an adversarial environment requires us to anticipate that our opponent will try to cause machine learning to fail in many ways. In this paper, we discuss both a theoretical framework for understanding adversarial machine learning, and then discuss a number of specific examples illustrating how these techniques succeed or fail.

Advances in computing capabilities have made online statistical machine learning a practical and useful tool for solving large-scale decision-making problems in many systems and networking domains, including spam filtering, network intrusion detection, and virus detection [36, 45, 60]. In these domains, a machine learning algorithm, such as a Bayesian learner or a Support Vector Machine (SVM) [14], is typically periodically retrained on new input data.

Unfortunately, sophisticated adversaries are well aware that online machine learning is being applied and we have substantial evidence that they frequently attempt to break many of the assumptions that practitioners make (*e.g.*, data has various weak stochastic properties; independence; a stationary data distribution).

The lack of stationarity provides ample opportunity for mischief during training (including periodic re-training) and classification stages. In many cases, the adversary is able to poison the learner's classifications, often in a highly targeted manner. For instance, an adversary can craft input data that has similar feature properties to normal data (*e.g.*, creating a spam message that appears to be non-spam to the learner), or they exhibit Byzantine behaviors by crafting input data that, when retrained on, causes the learner to learn an incorrect decision-making function. These sophisticated adversaries are patient and adapt their behaviors to achieve various goals, such as avoiding detection of attacks, causing benign input to be classified as attack input, launching focused or targeted attacks, or searching a classifier to find blind-spots in the algorithm.

Adversarial machine learning is the design of machine learning algorithms that can resist these sophisticated attacks, and the study of the capabilities and limitations of attackers. In this paper, we: give a taxonomy for classifying attacks against online machine learning algorithms; discuss

[*]This paper expands upon J. D. Tygar's invited talk at AISec 2010 on Adversarial Machine Learning describing the SecML project at UC Berkeley, and includes material from many of our collaborators. We kindly thank this year's AISec organizers for allowing us to present this paper.

AISec'11, October 21, 2011, Chicago, Illinois, USA.

application-specific factors that limit an adversary's capabilities; introduce two models for modeling an adversary's capabilities; explore the limits of an adversary's knowledge about the algorithm, feature space, training, and input data; explore vulnerabilities in machine learning algorithms; discuss countermeasures against attacks; introduce the evasion challenge; and discuss privacy-preserving learning techniques.

2. TAXONOMY

In prior work [2], we introduced a qualitative taxonomy of attacks against a machine learning system, which has since been extended to propose a framework for quantitatively evaluating security threats [37]. Our taxonomy categorizes an attack based on the following three properties:

Influence

- **Causative -** *Causative* attacks alter the training process through influence over the training data.
- **Exploratory -** *Exploratory* attacks do not alter the training process but use other techniques, such as probing the detector, to discover information about it or its training data.

Security violation

- **Integrity -** *Integrity* attacks result in intrusion points being classified as normal (false negatives).
- **Availability -** *Availability* attacks cause so many classification errors, both false negatives and false positives, that the system becomes effectively unusable.
- **Privacy -** In a *privacy* violation, the adversary obtains information from the learner, compromising the secrecy or privacy of the system's users.

Specificity (a continuous spectrum)

- **Targeted -** In a *targeted* attack, the focus is on a single or small set of target points.
- **Indiscriminate -** An *indiscriminate* adversary has a more flexible goal that involves a very general class of points, such as "any false negative."

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); (b) creating a denial of service event in which benign instances are incorrectly filtered as false positives (an *availability* violation); or (c) using the filter's responses to infer confidential information used in the learning process (a *privacy* violation). *Privacy* violations were not originally captured in [2]; and since they are qualitatively different to *integrity* and *availability* violations we discuss them separately in Section 5.

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, can potentially be a spectrum of choices, but for simplicity, we will categorize attacks and defenses into these groupings.

2.1 Game Specification and Interpretation

We model secure learning systems as a game between an attacker and a defender—the attacker manipulates data to mis-train or evade a learning algorithm chosen by the defender to thwart the attacker's objective. This game can be formalized in terms of a learning algorithm H; and the attacker's data corruption strategies $A^{(\mathrm{train})}$ and $A^{(\mathrm{eval})}$. The resulting game can be described as follows:[1]

1. **Defender** Choose learning algorithm H for selecting hypotheses based on observed data
2. **Attacker** Choose attack procedures $A^{(\mathrm{train})}$ and $A^{(\mathrm{eval})}$ (potentially with knowledge of H)
3. Learning:
   - Obtain dataset $\mathbb{D}^{(\mathrm{train})}$ with contamination from $A^{(\mathrm{train})}$
   - Learn hypothesis: $f \leftarrow H\left(\mathbb{D}^{(\mathrm{train})}\right)$
4. Evaluation:
   - Obtain dataset $\mathbb{D}^{(\mathrm{eval})}$ with contamination from $A^{(\mathrm{eval})}$
   - Compare predictions $f(x)$ to y for each data point $(x, y) \in \mathbb{D}^{(\mathrm{eval})}$

This game structure describes the fundamental interactions between the defender and attacker in choosing H, $A^{(\mathrm{train})}$ and $A^{(\mathrm{eval})}$; these steps are depicted in Figure 1. The defender chooses H to select hypotheses that predict well regardless of $A^{(\mathrm{train})}$ and $A^{(\mathrm{eval})}$, while the attacker chooses $A^{(\mathrm{train})}$ and $A^{(\mathrm{eval})}$ to produce poor predictions. The characteristics specified by the taxonomy's axes further specify some aspects of this game. The INFLUENCE axis determines the structure of the game and the legal moves that each player can make. In *exploratory* attacks, the procedure $A^{(\mathrm{train})}$ is not used in the game, and thereby the attacker only influences $\mathbb{D}^{(\mathrm{eval})}$. Meanwhile, in the *causative* game the attacker also has indirect influence on f through his choice of $A^{(\mathrm{train})}$. The SPECIFICITY and SECURITY VIOLATION axes of the taxonomy determine which instances the adversary would like to have misclassified during the evaluation phase. In an *integrity* attack, the attacker desires false negatives and therefore will use $A^{(\mathrm{train})}$ and/or $A^{(\mathrm{eval})}$ to create or discover false negatives, whereas in an *availability*, the attacker will also try to create or exploit false positives. Finally, in a *targeted* attack the attacker only cares about the predictions for a small number of instances, while an *indiscriminate* attacker cares about prediction for a broad range of instances.

3. CAUSATIVE ATTACKS

We first discuss *causative* attacks, in which the adversary influences the training data. Most importantly, the adversary in a *causative* attack alters the training data with a transformation $A^{(\mathrm{train})}$. The attacker may have various types of influence over this data, ranging from arbitrary

[1] Note that, since the game described here is batch training, an adaptive procedure $A^{(\mathrm{train})}$ is unnecessary unless the distribution $P_{\mathcal{Z}}^{(\mathrm{train})}$ is non-stationary in which case periodic retraining may be desirable. We return to this issue in Section 3.4.2.

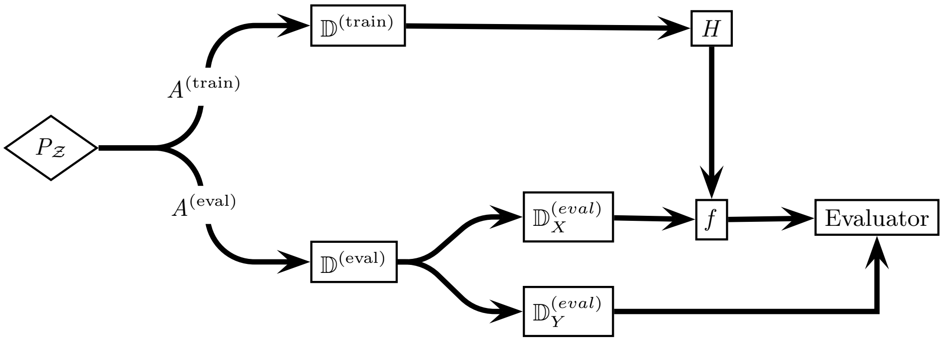

Figure 1: Diagram of an attack against a learning system where $P_{\mathcal{Z}}$ is the data's true distribution, $A^{(\text{train})}$ and $A^{(\text{eval})}$ are adversary's attack procedures, $\mathbb{D}^{(\text{train})}$ and $\mathbb{D}^{(\text{eval})}$ are the training and test datasets, H is the learning algorithm, and f is the hypothesis it learns from the training data. The hypothesis is evaluated on the test data by comparing its prediction $f(x)$ to the true label y for each $(x, y) \in \mathbb{D}^{(\text{eval})}$.

control over some fraction of training instances to a biasing influence over some aspect of data production; these details depend largely on the application as we discuss below. Regardless, the attacker uses his influence to mislead the learner causing it to produce a bad classifier, which the adversary subsequently exploits during evaluation. As in *exploratory* attacks (see Section 4), a causative adversary also can use $A^{(\text{eval})}$ to alter the evaluation data. Naturally, a *causative* adversary can coordinate $A^{(\text{train})}$ and $A^{(\text{eval})}$ to best achieve his objective, although in some *causative* attacks, the adversary may only be able to exert control over the training data (*e.g.*, the attacker in the case study below can not control the evaluation non-spam messages).

Several researchers have studied *causative* attacks. Newsome *et al.* [52] constructed *causative* attacks against the Polygraph virus detector, a polymorphic-virus detector that learns virus signatures using both a conjunction learner and a naive-Bayes-like learner. The *correlated outlier* attack, a *causative availability* attack, targets the naive-Bayes-like component of this detector by adding spurious features to positive training instances, causing the filter to block benign traffic with those features. Newsome *et al.* also developed a *causative integrity* attack against the conjunction learner component of Polygraph. This *red herring* attacks introduces spurious features along with their payload; once the learner has constructed its signature from this flawed data, the spurious features are discarded to avoid subsequent detection. Venkataraman *et al.* also present lower bounds for learning worm signatures based on red herring attacks [63]. Chung and Mok also developed *allergy* attacks against the Autograph worm signature generation system [12, 13]. Autograph operates in two phases. First, it identifies infected nodes based on behavioral patterns, in particular scanning behavior. Second, it observes traffic from the suspect 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; thus, this is a *causative availability* attack. We now describe two attacks that we studied in prior work [47, 48, 57], and subsequently we discuss general perspectives and guidelines we developed for analyzing learning systems that augment our taxonomy.

Case Study: SpamBayes

SpamBayes is a content-based statistical spam filter that classifies email using token counts [55]. SpamBayes computes a spam score for each token in the training corpus based on its occurrence in spam and non-spam emails; this score is motivated as a smoothed estimate of the posterior probability that an email containing that token is spam. The filter computes a message's overall spam score based on the assumption that the token scores are independent and then it applies Fisher's method [26] for combining significance tests to determine whether the email's tokens are sufficiently indicative of one class or the other. The message score is compared against two thresholds to select the label *spam*, *ham* (*i.e.*, non-spam), or *unsure*.

In analyzing the vulnerabilities of SpamBayes, we were motivated by the taxonomy of attacks. Known real-world attacks that spammers use against deployed spam filters tend to be *exploratory integrity* attacks: either the spammer obfuscates the especially spam-like content of a spam email or he includes content not indicative of spam. Both tactics aim to get the modified message into the victim's inbox. This category of attack has been studied in detail in the literature [16, 39, 40, 65]. However, we found the study of *causative* attacks more compelling. In particular, we demonstrated a *causative availability* attack that created a powerful denial of service [47]; *i.e.*, if many legitimate messages are filtered by the user's spam filter, the user is likely to disable the filter and therefore see the spammer's advertisements. Alternatively, an unscrupulous business owner may wish to use spam filter denial of service to prevent a competitor from receiving email orders from potential customers.

We designed two types of *causative availability* attacks, one *indiscriminate* and the other *targeted*, against SpamBayes. The first is an *indiscriminate dictionary* attack, in which the attacker sends attack messages that contain a very large set of tokens—the attack's *dictionary*. After training on these attack messages, the victim's spam filter will have a higher spam score for every token in the dictionary. As a result, future legitimate email is more likely to be marked

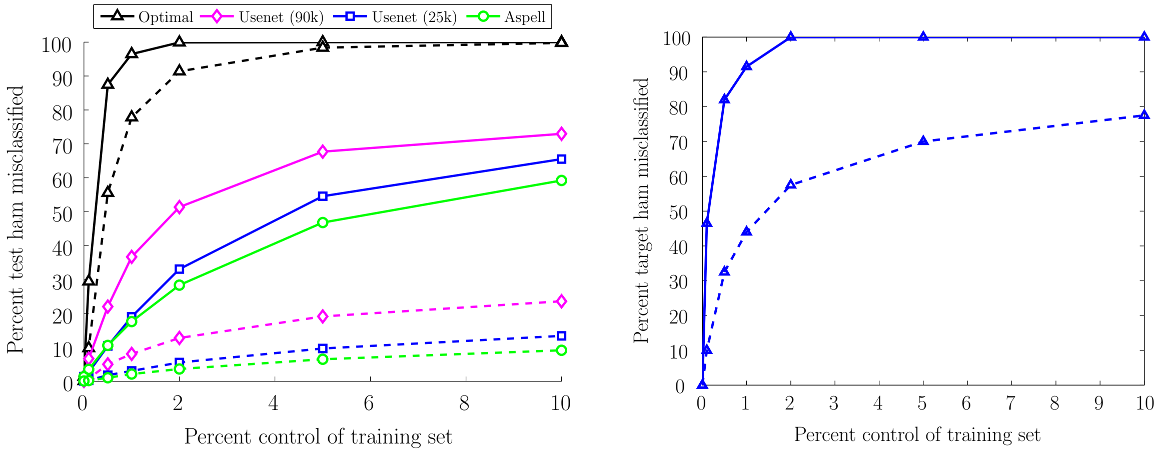

Figure 2: Effect of the dictionary and focused attacks. We plot percent of ham classified as *spam* (dashed lines) and as *unsure* or *spam* (solid lines) against percent of the training set contaminated. *Left:* Three dictionary attacks on an initial training set of 10,000 messages (50% spam). We show the optimal attack (black △), the Usenet dictionary attack with 90,000 words (magenta ⋄), the Usenet dictionary attack with 25,000 words (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).

as *spam* since it will contain many tokens from that lexicon (see Section 3.1). For instance, if only the victim's language is known by the attacker, the attack dictionary can be that language's entire lexicon. A refinement of this attack instead uses a token source with a distribution closer to the victim's true email distribution (see Section 3.3 for more on attacker knowledge). Using the most common tokens may allow the attacker to send smaller emails without losing much effectiveness. However, there is an inherent trade-off in choosing tokens: rare tokens are the most vulnerable since their scores have less support and will change quickly with few attack emails but they are also less likely to appear in future messages, diluting their usefulness. We discuss this trade-off in Section 3.1.3.

Our second attack is a *targeted* attack—the attacker has some knowledge of a specific legitimate email he targets to be incorrectly filtered. If the attacker has exact knowledge of the target email, placing all of its tokens in attack emails produces an optimal targeted attack. Realistically, though, the attacker only has partial knowledge about the target email and can guess only some of its tokens to include in attack emails (see Section 3.3). We modeled this knowledge by letting the attacker know a certain fraction of tokens from the target email, which are included in the attack message. The attacker constructs attack email that contain words likely to occur in the target email; *i.e.*, the tokens known by the attacker. The attack email may also include additional tokens added by the attacker to obfuscate the attack message's intent. When SpamBayes trains on the resulting attack email, the spam scores of the targeted tokens generally increase and the target message is more likely to be filtered as *spam*. This is the focused attack.

In our prior work [47], we presented results demonstrating the effectiveness of both dictionary and focused attacks, reproduced in Figure 2. These graphs depict the effectiveness of dictionary attacks (leftmost figure) and focused attacks (rightmost figure) in causing misclassifications in terms of the percent of attack messages in the training set (see Section 3.2 for discussion of the attacker's capabilities). Notice that since SpamBayes has three predictions (*ham*, *unsure*, and *spam*), we plot the percentage of messages misclassified as *spam* (dashed lines) and either as *unsure* or *spam*. As the figure demonstrates, these attacks are highly effective with a small percentage of contamination.

Case Study: Anomalous Traffic Detection

Adversaries can use *causative* attacks to not only disrupt normal user activity but also to achieve evasion by causing the detector to have many false negatives through an *integrity* attack. In doing so, such adversaries can reduce the risk that their malicious activities are detected.

Here we reflect on our study of the subspace anomaly detection methods for detecting network-wide anomalies such as denial-of-service (DoS) attacks. In this study, we showed that by injecting crafty chaff into the network during training, the detector can be poisoned so that it is unable to effectively detect a subsequent *DoS* attack. The detector we analyze was first proposed as a method for identifying volume anomalies in a backbone network based on the Principal Component Analysis (PCA) dimensionality reduction technique [36]. While their subspace-based method is able to successfully detect *DoS* attacks in network traffic, it assumes the detector is trained on non-malicious data (in an unsupervised fashion under the setting of *anomaly detection*). Instead, we considered an adversary who knows that an ISP is using the subspace-based anomaly detector and attempts to evade it by proactively poisoning its training data.

The goal of the adversary we considered was to circumvent detection by poisoning the training data; *i.e.*, an *integrity* goal to increase the detector's false negative rate, which corresponded to the evasion success rate of the attacker's sub-

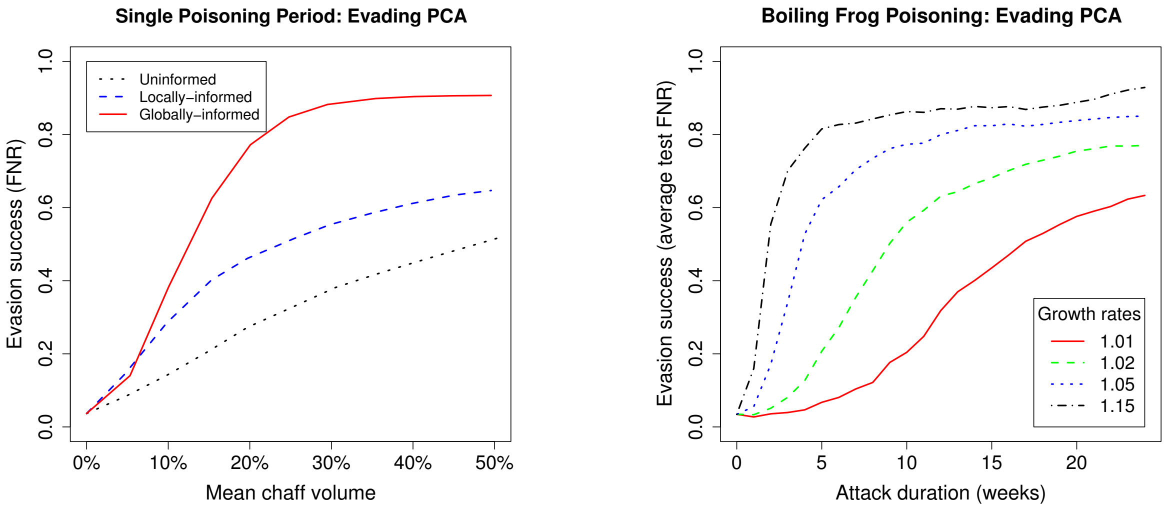

Figure 3: Effect of poisoning attacks on the PCA-based detector [36]. *Left:* Evasion success of PCA versus relative chaff volume under Single-Training Period poisoning attacks using three chaff methods: uninformed (dotted black line) locally-informed (dashed blue line) and globally-informed (solid red line). *Right:* Evasion success of PCA under Boiling Frog poisoning attacks in terms of the average FNR after each successive week of locally-informed poisoning for four different poisoning schedules (*i.e.*, a weekly geometric increase in the size of the poisoning by factors 1.01, 1.02, 1.05, and 1.15 respectively). More aggressive schedules (*e.g.*, growth rates of 1.05 and 1.15) significantly increase the FNR within a few weeks while less aggressive schedules take many weeks to achieve the same result but are more stealthy in doing so.

sequent *DoS* attack. When trained on this poisoned data, the detector learned a distorted set of principal components that are unable to effectively discern these *DoS* attacks—a *targeted* attack. Because PCA estimates the data's principal subspace solely on the covariance of the link traffic, we explored poisoning schemes that add *chaff* (additional traffic) into the network along the flow targeted by the attacker to systematically increase the targeted flow's variance. In so doing, the attacker caused the estimated subspace to unduly shift toward the target flow making large-volume events along that flow less detectable. We considered three general categories of attacks based on the attacker's capabilities: uninformed attacks, locally-informed attacks, and globally-informed attacks. Each of these reflect different levels of knowledge and resources available to the attacker; see Section 3.3 and 3.1 for more detailed discussion of these models.

In the above attacks, chaff was designed to impact a single period (one week) in the training cycle of the detector, but we also considered the possibility of episodic poisoning which are carried out over multiple weeks of retraining the subspace detector; see Section 3.4.2 for further discussion of iterative retraining. Multi-week poisoning strategies vary the attack according to the time horizon over which they are carried out. As with single-week attacks, during each week the adversary inserts chaff along the target flow throughout the training period according to his poisoning strategy. However, in the multi-week attack the adversary increases the total amount of chaff used during each subsequent week according to a poisoning schedule. This poisons the model over several weeks by initially adding small amounts of chaff and increasing the chaff quantities each week so that the detector is gradually acclimated to chaff and fails to adequately identify the eventually large amount of poisoning. We call this type of episodic poisoning the Boiling Frog poisoning method after the folk tale that one can boil a frog by slowly increasing the water temperature over time. The goal of Boiling Frog poisoning is to gradually rotate the normal subspace, injecting low levels of chaff relative to the previous week's traffic levels so that PCA's rejection rates stay low and a large portion of the present week's poisoned traffic matrix is trained on. Although PCA is retrained each week, the training data will include some events not caught by the previous week's detector. Thus, more malicious training data will accumulate each successive week as the PCA subspace is gradually shifted. This process continues until the week of the *DoS* attack, when the adversary stops injecting chaff and executes their desired *DoS*.

In our prior work [57], we empirically demonstrated our attacks against PCA and in Figure 3 we reproduce the results of our experiments. These graphs depict the effectiveness of the Single-Training Period (leftmost figure) and Boiling Frog attacks (rightmost figure) in causing false negatives in terms of the percent of average increase in the mean link rates due to chaff (see Section 3.2 for discussion of the attacker's capabilities) and the length of the attack duration, respectively. For the Boiling Frog attacks, we assumed that the PCA-subspace method is retrained on a weekly basis using the traffic observed in the previous week to retrain the detector at the beginning of the new week. Further, we sanitize the data from the prior week before retraining so that all detected anomalies are removed from the data. As the Figure 3 demonstrates, these attacks cause high rates of misdetection with relatively small increases in the volume of traffic: *e.g.*, a locally-informed attacker can increase his evasion success to 28% from the baseline of 3.67% via a 10% average increase in the mean link rates due to chaff.

3.1 Incorporating Application-Specific Factors

One type of adversarial limitation we consider are limits on how an adversary can alter data points in terms of each *feature*. Features represent different aspects of the state of the world and have various degrees of vulnerability to attack. Some features can be arbitrarily changed by the adversary, but others may have stochastic aspects that the adversary cannot completely control, and some features may not be alterable at all. For instance, in sending an email, the adversary can completely control the content of the message but cannot completely determine the routing of the message or its arrival time. Further, this adversary has no control over meta-information that is added to the message by mail relays while the message is en route. Providing an accurate description of the adversary's control over the features is essential and we discuss it further in the following section.

3.1.1 Domain Limitations

The first consideration are limitations on the adversary that arise from the application domain itself. These include limits on how the adversary can interact with the application and what kinds of data are realistic for the adversary to manipulate. For SpamBayes, the usual usage scenario is that this detector is intended to be used for individuals or small organizations to filter their email. In such a scenario, it is difficult to believe that filter's users would intentionally mislabel messages to corrupt the filter.[2] Thus, to obtain faulty labels, the adversary would have to rely on being able to fool users to mislabel their messages. However, it is not easy to trick users to mislabel spam messages as non-spam. It is more realistic that a user could be tricked into mislabeling non-spam messages as spam (*e.g.*, by scrambling the legitimate message and adding spam-like elements). From this observation, we concluded that *integrity* attacks requiring mislabeled non-spam were unrealistic and instead developed our *availability* attack using *tricky* messages that a user would probably mislabel as *spam* even if they were not a traditional advertising pitch.

The intended use-case for the PCA-based anomaly detector was completely different. It was intended to operate along the routing infrastructure of a backbone network. For this scope, the detector is not given any label information at all; all training data is unlabeled and anomalous data is identified by the detector only after it is trained. Thus, for this application, the attacker has a broad ability to manipulate data so that both *integrity* and *availability* attacks were potentially feasible.

3.1.2 Contrasting Feature Spaces

A second consideration are limitations imposed on the adversary by the space of features used by the learning algorithm. In many learning algorithms, data is represented in a feature space, in which each feature captures a relevant aspect of data points for the learning task at hand. For spam detection, it is common for each feature to represent whether or not a particular word or token appears in the message; this is the feature space used by SpamBayes. In the network volume anomaly detection application, the feature space used represented the volume of traffic along each link within the network during a specific period of time. These differences in feature space representations profoundly impact the set of actions available to the adversary and thus significantly shape how attacks can be carried out.

SpamBayes has a large (one could say infinite) feature space of possible tokens that can occur in an email message. However, while there are many features, the influence the adversary has on each is severely limited. In fact, all the adversary can decide is whether or not to include that token in his attack messages; this bounded form of influence ensures that the adversary cannot simply manipulate a single feature to have a large impact on the detector's performance. That is, it was not possible for a small number of emails controlled by the adversary to only use a small number of features in order to damage the filter. However, the fact that a single email could influence many features simultaneously (and independently) meant that attacks messages could be constructed with undue influence on the detector as a whole. This motivated the dictionary attacks discussed above.

The domain of the PCA-based anomaly detector had very different characteristics. It only used a small number of features to represent each time-based data point; a single feature for each link in the Abilene backbone network for a total of 54 features. However, unlike the binary features in the spam application, each of these features was real-valued. This meant that the adversary did not need to influence a large number of features, but instead he could dramatically alter a small number of features. By doing so, the adversary could have nearly unlimited control of a single data point but a few restrictions were necessary. First, in this domain, there were physical capacities in the network links that could not be exceeded. We also assumed the adversary does not have control over existing traffic (*i.e.*, he cannot delay or discard traffic). Similarly, the adversary cannot falsify SNMP reports to PCA because stealth is a major goal for this attacker—he does not want his DoS attack or his poisoning to be detected until the DoS attack has successfully been executed. As such, we limited the attacks to only add spurious traffic to the network thereby further limiting the adversary's capabilities.

3.1.3 Contrasting Data Distributions

Another application-specific aspect of a threat to a learning algorithm is the underlying properties of the data. The data's distribution can profoundly impact not only the performance of the learning algorithm but also its vulnerability. The data's distribution may have properties that conflict with the learner's assumptions.

We found that SpamBayes was vulnerable in part due to the inherent sparsity of emails; that is, the fact that most messages only contain a small subset of the possible set of words and tokens that can appear in a message. This is a well-known property of most documents and, moreover, it is well known that the words in emails tend to follow a Zipf distribution; that is, the frequency that a word occurs in a document is approximately inversely proportional to the word's popularity rank. These properties couple with SpamBayes' independence assumption (see next section) to have the following consequences: (a) popular tokens occur often in the corpus and thus have stable probability estimates that cannot easily be changed (b) most tokens, however, occur rarely and thus have very low support in the

[2] For web-based mailers, the situation changes dramatically since many users of these services may be malicious and may, in fact, intentionally mislabel messages. This potential for attacks by the users can be viewed as a type of *insider threat* which is a topic for future work.

non-attack data so they were easily influenced by the poison messages. SpamBayes benefits from the first property because the more often a word appears in good training data, the less vulnerable that word is to poisoning. Further, rare words are vulnerable but will not be likely to appear in future messages. However, while it is unlikely that a particular rare word would appear in a new message, it is still quite likely that several rare words will appear in the message—the Zipf distribution is long-tailed meaning that the tail of it has significant mass. Thus, by poisoning many (or all) rare words, the attack against SpamBayes was spectacularly successful.

The data in the network volume anomaly detection algorithm also had particular properties that an adversary would be able to exploit. Unlike spam, the data was not sparse in link space—the Abilene backbone network carried large volumes of traffic across almost all of it's link in even unit of time. However, it is well-established that there are significant size discrepancies in the flow space of the network; *i.e.*, there are a small number of end-to-end flows in the network that account for most of its traffic. These flows (nicknamed 'elephant' flows) dwarfed the numerous small flows (nicknamed 'mouse' flows). Moreover, while elephant flows always carried traffic, mouse flows tended to be spiky with almost no traffic most of the time and occasional spikes of large amounts of data. As with SpamBayes, attacks against the PCA-based detector exploit this distributional property.

3.2 Modeling Attacker Capabilities

Two elements are critical to define a model for the adversary: his motivation/objective and his capabilities. The taxonomy partially describes both, but here we delve further into how one can describe the capabilities of an attacker in a *causative* attack. It is critical to define how the adversary can alter data to mislead or evade the classifier. For this purpose, we need to model the restrictions on the adversary and justify these restrictions for a particular domain.

3.2.1 Corruption Models

Here, we outline two common models for adversarial corruption, and we describe how the adversary is limited within each. The first model assumes the adversary has unlimited control of a small fraction of the data; *i.e.*, the adversary is restricted to only modify a limited amount of data but can alter those data points arbitrarily. We call this an *insertion model* because, in this scenario, the adversary crafts a small number of attack instances and *inserts* them into the dataset for training or evaluation (or perhaps replaces existing data points). For example, in the example of a spam filter, the adversary (spammer) can create any arbitrary message for their attack but he is limited in the number of attack messages he can inject; thus, the spammer's attack on the spam filter can be analyzed in terms of how many messages are required for the attack to be effective. For this reason, the insertion model was appropriate in analyzing attacks on the SpamBayes.

The second corruption model instead assumes that the adversary can alter any (or all) of the data points in the data set but is limited in the degree of alteration; *i.e.*, an *alteration model*. For example, to attack a detector that is monitoring network traffic volumes over windows of time, the adversary can add or remove traffic within the network but only can make a limited degree of alteration. Such an adversary cannot insert new data since each data point corresponds to a time slice and the adversary cannot arbitrarily control any single data point since other actors are also creating traffic in the network; thus, this is the model we used to analyze attacks on the PCA-subspace detector. Here, the adversary is restricted by the total amount of alteration they make, and so the effectiveness of his attack can be analyzed in terms of the size of alteration required to achieve the attacker's objective.

3.2.2 Class Limitations

A second limitation on attackers involves which parts of the data the adversary is allowed to alter—the positive (malicious) class, the negative (benign) class, or both. Usually, attackers external to the system are only able to create malicious data and so they are limited to only manipulating positive instances. This is the model we use throughout this text. However, there is also an alternative threat that *insiders* could attack a learning system by altering negative instances; a lucrative direction for future work.

3.3 Attacker Knowledge

The final aspect of the attacker that we model is the amount of information the attacker has about the learning system. Generally, the adversary has degrees of information about three components of learner: its learning algorithm, its feature space, and its data. As with the attacker's capabilities, it is critical to make a reasonable model of the attacker's information. The relevant guideline for assumptions about the adversary is *Kerckhoffs' Principle* [34]; *i.e.*, the security of a system should not rely on unrealistic expectations of secrecy. An over-dependence on secrets to provide security is dangerous because if these secrets are exposed the security of the system is immediately compromised. Ideally, secure systems should make minimal assumptions about what can realistically be kept secret from a potential attacker. On the other hand, if the model gives the adversary an unrealistic degree of information, our model may be overly pessimistic; *e.g.*, an omnipotent adversary who completely knows the algorithm, feature space, and data can exactly construct the learned classifier and design optimal attacks accordingly. Thus, it is necessary to carefully consider what a realistic adversary can know about a learning algorithm and to quantify the effects of that information.

With these constraints in mind, we generally 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. We examine different degrees of the attacker's knowledge and assess how much he gains from different sources of potential information.

3.3.1 Knowledge about the Learning Algorithm

In our analyses of real-world learning algorithms, we have generally assumed that the adversary will know the exact learning algorithm that they target. For our case studies, the algorithms were public and, for SpamBayes, the source code was freely available. However, we generally believe that system designers should assume that their learning algorithm is known by the adversary; just as the encryption algorithms in cryptography, are generally assumed to be known.

One potential source of secret information about the algorithm is a secret random component (*e.g.*, a randomized initial state). Such a component could be an important secret if it truly makes a random contribution (*e.g.*, many random initial states may yield the same decision function). The authors are not aware of any strong guarantees provided by this sort of randomization for the security domain although randomization is discussed below in the context of privacy-preserving learning.

3.3.2 Knowledge about the Feature Space

The feature space is potentially a component of the learning algorithm that may be kept secret (although in both SpamBayes and the PCA-based network anomaly detector, the feature space was published). Specialized features constructed for a specific learning problem or relevant discriminant features found by a feature selection algorithm may not be known or inferable by the adversary. However, the system designer should carefully assess the viability of this assumptions since many features are widely used for some learning tasks (*e.g.*, unigram features in document classification) and specially selected features may be approximated with simpler features.

3.3.3 Knowledge of Data

The final component of the attacker's knowledge is his knowledge about the training and evaluation data used by the algorithm. Stronger restrictions on this component are more reasonable because, in many scenarios, strong protections on user data are a separate component of the system. However, a system designer should consider ways in which the adversary can learn about the systems data and how specific that knowledge is. For instance, part of the data may be available to the adversary because of actions the adversary makes outside the system (*e.g.*, a spammer can send spam to a particular spam filter) or because the adversary is an insider. Adversaries also can have a degree of global information about the data although this tends to be less specific (*e.g.*, an adversary may have distributional information about the words commonly used in emails or the length of messages).

In our SpamBayes case study, we considered an adversary with several different degrees of information about the data. When the attacker only has vague information about email word characteristics (*e.g.*, the language of the victim), we showed that a broad dictionary attack can render the spam filter unusable, causing the victim to disable the filter. With more detailed information about the email word distribution, the attacker can select a smaller dictionary of high-value words that are at least as effective. Finally, when the attacker wants to prevent a victim from seeing particular emails and has some information about those emails, the attacker can target them with a *focused attack* that specifically targets the words that are likely to be used in the targeted message.

Similarly in our study of PCA-based anomaly detectors, we considered poisoning strategies in which the attacker has various potential levels of information at his disposal. The weakest attacker is one that knows nothing about the traffic flows, and adds chaff randomly (called an *uninformed* attack). Alternatively, a partially-informed attacker knows the current volume of traffic at a compromised ingress link that he intends to inject chaff on. We call this type of poisoning a *locally-informed* attack because this adversary only observes the local state of traffic at the ingress PoP of the attack. In a third scenario, the attacker is *globally-informed* because his global view over the network enables him to know the traffic levels on all network links and this attacker has knowledge of all future link traffic. Although global information is impossible to achieve, we studied this scenario to better understand the limits of variance injection poisoning schemes.

3.4 Identifying Learning Vulnerabilities

Here we consider the mechanism by which the adversary attacks learners—the vulnerabilities of learning algorithms. The first part of an algorithm's vulnerability lies in the assumptions it makes about the training data. The second part arises from retraining procedures, which can be used by the adversary to amplify a weak attack into a much stronger one by coordinating over many retraining iterations.

3.4.1 Learning Assumptions

Every learning algorithm must make some assumptions about the training data and the space of possible hypotheses to make learning tractable [46]. However, these modeling assumptions can also lead to vulnerabilities to adversarial corruption. There are two types of learning assumptions: assumptions made by the learning model and assumptions made by the training algorithm.

Modelling Assumptions

Assumptions made by the learning model are *intrinsic* to the model itself; *i.e.*, they are assumptions involving the properties of the classifier and its representation of the data. Common modelling assumptions include data linearity (the learning task can be represented by a linear function), separability (there exists some function that separates the data into distinct classes), and feature independence (each feature of an object is an independent indicator of a latent variable that represents the true class of the overall object).

The first modeling vulnerability of SpamBayes comes from its assumption that the data and tokens are independent, for which each token score is estimated based solely on the presence of that token in spam and non-spam messages. The second vulnerability comes from its assumption that only tokens that occur in a message contribute to its label. While there is some intuition behind the latter assumption, in this model, it causes rare tokens to have little support so that their scores can be easily changed (as discussed above). Ultimately, these two vulnerabilities lead to a family of dictionary attacks discussed above.

In the PCA-based detector, the modeling assumption is a linearity assumption that normal data is well-represented by low-dimensional subspace in the link space of the algorithm (with a small residual component). This assumption is empirically validated [36], but can be violated by a clever adversary as discussed below. In this case, the attack arises more from an assumption made by the training algorithm since most of the data does obey the low-dimensionality assumption but the learning algorithm over-leverages it.

Data Assumption used in Training

Many vulnerabilities arise from the distributional assumptions made in the learning procedure about the data used during the learning and evaluation phases. Often these assumptions are used to construct more efficient learning procedures that require less data but can result in vulnerabilities when violated. To defend against or patch these vulnerabilities, an alternative learning algorithm can be selected that makes weaker assumptions although it may be less efficient; *i.e.*, there is a fundamental trade-off between the efficiency of a procedure and its robustness to violations in its assumptions.

Many learning methods make a *stationarity* assumption; *i.e.*, the training data and evaluation data are drawn from the same distribution. 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. Violations of the stationarity assumption comprise the fundamental vulnerability of learning algorithms, and thus, analyzing and strengthening learning methods to withstand or mitigate violations of the stationarity assumption is the crux of the secure learning problem.

Another common assumption is that each data point is independent and identically distributed (i.i.d.); clearly, this assumption is violated if the adversary can coordinate to create correlation between data points or if the adversarial data comes from an alternative distribution. Both the SpamBayes and PCA-based anomaly detector assume their data is i.i.d. and both are vulnerable because of it. In SpamBayes, the background data is not altered, but the attack data that is introduced is designed to come from a different distribution (recall the data sparsity in SpamBayes discussed above). Similarly, our attacks on the PCA-base detector violate the i.i.d. assumption by introducing a perturbation to a subset of the data that systematically shifts these points onto a second subspace; thus the data is no longer identically distributed. Moreover, this perturbation can occur (in locally-informed and globally informed attacks) in a data-dependent fashion also violating independence.

3.4.2 Iterative Retraining

Iterative retraining is perhaps the most desirable but also least well-understood aspect of learning in adversarial environments. Many detectors in such environments require periodic retraining because of non-stationarity in *regular* data causing its distribution to gradually shift over time. Retraining can be also be an effective defense against adversaries to counter their ability to learn about and adapt to a detector. Finally, iterative learning has an extensive theory for combining classifiers even under strong adversarial settings [9]. Simultaneously, retraining, if applied improperly, can be exploited by the adversary potentially to amplify the impact of weak attacks over many iterations.

There is little doubt that past experience could be tremendously valuable for improving classifiers, but how can the past be used effectively and safely? If the past is not used at all, every iteration of retraining is seemingly independent, but even in this case, past models may subtlety influence future models. Most usually, some of the data used for the training the next iteration of models is selected or labeled by the last learned model—this small influence can be used in attacks such as the Boiling Frog attack against the PCA-based detector to make a small attack far more effective as discussed above. More subtlety, the behavior of the classifier may cause a user to pay more attention to some data points than others and thus impact the subsequent retraining process. For instance, in the case of SpamBayes, the mistakes SpamBayes makes may cause the user to specifically identify and retrain on attack messages when these messages appear with his normal mail. However, most users are less likely to notice messages that inadvertently mislabeled as spam.

If the past is wholly incorporated into the next generation of model, the advantage seems to lie with the learning model as we explored in prior work with retraining hypersphere anomaly detectors [49]. In that work, we showed that when all past data is used to update the mean of the hypersphere, the number of data points the adversary must control to shift the model is exponential in the desired distance. However, this comes at a price; because past data is never discarded, the model becomes increasingly less adaptive to normal changes in the data. Kloft and Laskov extended this model by considering more realistic policies for data aging and a more realistic setting for the attack [35]. However, the general task of retraining models in a secure fashion remains an open issue.

3.5 Defenses and Countermeasures

The final component in our discussion of *causative* attacks is methods for defending against them. Two general strategies for defense are to remove malicious data from the training set and to harden the learning algorithm against malicious training data. Below we discuss both of these techniques in the context of the SpamBayes and PCA-based detectors discussed above.

3.5.1 Data Sanitization

Insidious *causative* attacks make learning inherently more difficult. In many circumstances, data sanitization may be the only realistic mechanism to achieve acceptable performance. For example, for SpamBayes we explored such a sanitization technique called the *Reject On Negative Impact (RONI) defense* [48], a technique that measures the empirical effect of adding each training instance and discards instances that have a substantial negative impact on classification accuracy. To determine whether a candidate training instance is malicious or not, the defender trains a classifier on a base training set, then adds the candidate instance to the training set and trains a second classifier. The defender applies both classifiers to a *quiz set* of instances with known labels and measures the difference in accuracy between the two classifiers. If adding the candidate instance to the training set causes the resulting classifier to produce substantially more classification errors, the defender permanently removes the instance as detrimental in its effect.

The RONI defense rejects every single dictionary attack from any of the dictionaries (optimal, Aspell, and Usenet). In fact, the degree of change in misclassification rates for each dictionary message is greater than five standard deviations from the median, suggesting that these attacks are easily eliminated with only minor impact on the performance of the filter. When trained on $1,000$ uncensored messages, the resulting filter correctly classifies 98% of ham and 80% of the spam. After removing the messages rejected by the RONI defense, the resulting filter still correctly classifies 95% of ham and 87% of the spam.

3.5.2 Robust Learning

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 [30], it is assumed that the bulk of the data is generated from a known well-behaved model, but a fraction of the data comes from an unknown model (outliers)—to bound the effect of this unknown source it is assumed to be adversarial. Because the network data has many outlier events (both malicious and benign) we chose to replace the PCA-based detector with a more robust variant.

It is known that PCA can be strongly affected by outliers [54]. Instead of finding the principal components along directions that maximize variance, alternative PCA-like techniques find more robust components by maximizing alternative dispersion measures with desirable robustness properties. In particular, the PCA-Grid algorithm was proposed by Croux *et al.* as an efficient method for estimating directions that maximize the median absolute deviation (MAD) without under-estimating variance [15]. We adapt PCA-Grid for anomaly detection by combining the method with a robust Laplace cutoff threshold. Together, we refer to the method as Antidote. Because it builds on robust subspace estimates, this method substantially reduces the effect of outliers and is able to reject poisonous training data.

4. EXPLORATORY ATTACKS

The most frequently studied attacks are *exploratory integrity* attacks in which the adversary attempts to passively circumvent the learning mechanism to exploit blind spots in the learner that allow miscreant activities to go undetected. In an *exploratory integrity* attack, the attacker crafts intrusions so as to evade the classifier without direct influence over the classifier itself. Instead, attacks of this sort often attempt to systematically make the miscreant activity appear to be normal activity to the detector or obscure the miscreant activity's identifying characteristics. Some *exploratory integrity* attacks mimic statistical properties of the normal traffic to camouflage intrusions; *e.g.*, the attacker examines training data and the classifier, then crafts intrusion data. In the *exploratory* game, the attacker's move produces malicious instances in $\mathbb{D}^{(\text{eval})}$ that statistically resemble normal traffic in the training data $\mathbb{D}^{(\text{train})}$.

Exploratory integrity attacks have been extensively studied in attacks against intrusion detector systems (IDS) and spam filters. Fogla and Lee introduced *polymorphic blending attacks* that evade intrusion detectors using encryption techniques to make attacks statistically indistinguishable from normal traffic [27]. *Feature deletion attacks* instead specifically exclude high-value identifying features used by the detector [28]; this form of attack stresses the importance of proper feature selection as was also demonstrated empirically by [42] in their study of the behavior of intrusion detection systems on the DARPA/Lincoln Lab dataset. Tan *et al.* describe a *mimicry* attack against the `stide` sequence-based IDS [62]. 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. By exploiting the finite window size of the detector, this technique makes attack traffic indistinguishable from normal traffic for the detector. Independently, Wagner and Soto also developed mimicry attacks against pH, a sequence-based IDS [64]. Using the machinery of finite automata, they constructed a framework for testing whether an IDS is susceptible to mimicry for a particular exploit.

Adding or changing words in a spam message can allow it to bypass the filter. Like the attacks against an IDS above, these attacks all use both training data and information about the classifier to generate instances intended to bypass the filter. Studying these techniques was first suggested by John Graham-Cumming in his presentation *How to Beat an Adaptive Spam Filter* at the 2004 MIT Spam Conference, in which he presented a *Bayes vs. Bayes* attack that uses a second statistical spam filter to find good words based on feedback from the target filter. Several authors have further explored evasion techniques used by spammers and demonstrated attacks against spam filters using similar principles as those against IDSs as discussed above. Lowd and Meek and Wittel and Wu developed attacks against statistical spam filters that add *good words*, or words the filter considers indicative of non-spam, to spam emails [40, 65]. This *good word attack* makes 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. Finally, obfuscation of spam words (*i.e.*, changing characters in the word or the spelling of the word so it no longer recognized by the filter) is another popular technique for evading spam filters [38, 58].

4.1 Cost-based Evasion

Another vein of research into *exploratory integrity* attacks focuses on the costs incurred due to the adversary's evasive actions; *i.e.*, instances that evade detection may be less desirable to the adversary. By directly modelling adversarial cost, this work explicitly casts evasion as a problem, in which the adversary wants to evade detection but wants to do so using high-value instances (an assumption that was implicit in the other work discussed in this section). Dalvi *et al.* exploit these costs to develop a cost-sensitive game-theoretic classification defense that is able to successfully detect optimal evasion of the original classifier [16]. Using this game-theoretic approach, this technique preemptively patches the naive classifier's blind spots by constructing a modified classifier designed to detect optimally-modified instances. Subsequent game theoretic approaches to learning have extended this setting and solved for equilibria of the game [8, 32].

Cost models of the adversary also led to a theory for query-based near-optimal evasion of classifiers first presented by Lowd and Meek, in which they cast the difficulty of evading a classifier into a complexity problem [39]. They presented algorithms for an attacker to reverse engineer a classifier. The attacker seeks the lowest cost instance (for the attacker) that the classifier mislabels as a negative instance. In the next section we discuss our extension to this framework [50]. We generalized the theory of near-optimal evasion to a broader class of classifiers and demonstrated that the problem is easier than reverse-engineering approaches.

4.1.1 Near-Optimal Evasion

The near-optimal evasion problem formalizes the natural security setting for evasion. The problem abstracts the scenario of an adversary who wishes to launch a specific attack that is blocked by a classifier-based defense. The attacker has a limited number of probing opportunities after which he must send an attack as close as possible to his originally intended attack—a *near-optimal attack*. In the case of email spam, the spammer may originally have a message that will be detected as spam. He probes, finds a near-optimal message that evades the filter, and sends this message instead. In the case of an intruder, he has a preferred sequence of system calls that will be detected as intrusions. Again, he probes, then finds and executes a near-optimal sequence that evades the detector. With this framework in mind, we now clearly see the role of a defender: to provide a classifier that limits or resists near-optimal evasion. Practical implementation requires careful selection of costs and realistic bounds on the number of probes an adversary can perform. Resulting lower-bounds on the number of probes required for near-optimal evasion provide significant evidence of effective security.

The problem of *near-optimal evasion* (*i.e.*, finding a low cost negative instance with few queries) was introduced by Lowd and Meek [39]. We continued studying this problem by generalizing it to the family of convex-inducing classifier—classifiers that partition their instance space into two sets one of which is convex. The family of convex-inducing classifier is an important and natural set of classifiers to examine which includes the family of linear classifiers studied by Lowd and Meek as well as anomaly detection classifiers using bounded PCA [36], anomaly detection algorithms that use hyper-sphere boundaries [5], one-class classifiers that predict anomalies by thresholding the log-likelihood of a log-concave (or uni-modal) density function, and some quadratic classifiers. The family of convex-inducing classifier also includes more complicated bodies such as the countable intersection of halfspaces, cones, or balls.

In our work, we demonstrated that near-optimal evasion does not require complete reverse engineering of the classifier's internal state or decision boundary, but instead, only partial knowledge about its general structure [50]. The algorithm presented by Lowd and Meek for evading linear classifiers in a continuous domain reverse engineers the decision boundary by estimating the parameters of their separating hyperplane. The algorithms we presented for evading convex-inducing classifier do not require fully estimating the classifier's boundary (which is hard in the case of general convex bodies [53]) or the classifier's parameters (internal state). Instead, these algorithms directly search for a minimal cost-evading instance. These search algorithms require only polynomial-many queries, with one algorithm solving the linear case with better query complexity than the previously-published reverse-engineering technique. Finally, we also extended near-optimal evasion to general ℓ_p costs; *i.e.*, costs based on ℓ_p distances. We showed that the algorithms for ℓ_1 costs can also be extended to near-optimal evasion on ℓ_p costs, but are generally not efficient. However, in the cases when these algorithms are not efficient, we show that there is no efficient query-based algorithm.

There are a variety of ways to design countermeasures against exploratory attacks. For example, Biggio *et al.* promote randomized classifiers as a defense against exploratory evasion [4]. They propose the use of multiple classifier systems (MCSs), which are mainly used to improve classification accuracy in machine learning community, to improve the robustness of a classifier in adversarial environments. They construct a multiple classifier system using the bagging and random subspace methods, and conduct a series of experiments to evaluate their system in a real spam filtering task. Their results showed that, although their method did not improve the performance of a single classifier when they are not under attack, it was significantly more robust under attack. These results provide a sound motivation to the application of MCSs in adversarial classification tasks. However, it is not known if randomized classifiers have provably worse query complexities.

4.1.2 Real-World Evasion

While the cost-centric evasion framework presented by Lowd and Meek provides a formalization the near-optimal evasion problem, it fails to capture several important aspects of evasion in real-world settings. From the theory of near-optimal evasion, certain classes of learners can be evaded efficiently whereas others require a practically infeasible number of queries to achieve near-optimal evasion. However, real-world adversaries often do not require near-optimal cost evasive instances to be successful; it suffices for them to find any sufficiently low-cost instance that evades the detector. Understanding query-strategies for a real-world adversary requires incorporating real-world constraints that were relaxed or ignored in the theoretical version of this problem. Here, we summarize the challenges for real-world evasion.

Real-world near-optimal evasion is more difficult (*i.e.*, requires more queries) than is suggested by the theory because the theory simplifies the problem faced by the adversary. Even assuming that a real-world adversary can obtain query responses from the classifier, he cannot directly query it in the feature space of the classifier. Real-world adversaries must make their queries in the form of real-world objects like email that are subsequently mapped into the feature space of the classifier. Even if this mapping is known by the adversary, designing an object that maps to a specific desired query is itself difficult—there may be many objects that map to a single query and parts of the feature space may not correspond to any real-world object. Thus, future research on real-world evasion must address the question: *How can the feature mapping be inverted to design real-world instances to map to desired queries?*

Real-world evasion also differs dramatically from the near-optimal evasion setting in defining an *efficient* classifier. For a real-world adversary, even polynomially-many queries in the dimensionality of the feature space may not reasonable. For instance, if the dimensionality of the feature space is large (*e.g.*, hundreds of thousands of words in unigram models) the adversary may require the number of queries to be sub-linear, but for near-optimal evasion, this is not possible even for linear classifiers. However, real-world adversaries also need not be provably near-optimal. Near-optimality is a surrogate for adversary's true evasion objective: to use a small number of queries to find a negative instance with acceptably low-cost. Clearly, near-optimal evasion is sufficient to achieve this goal, but in real-world evasion, once a low-cost negative instance is located, the search can terminate. Thus, instead of quantifying the query complexity required for a family of classifiers, it is more relevant to quantify the query performance of an evasion algorithm for a fixed number of queries based on a target cost. This raises several questions: *What is the worst-case or expected reduction in cost for a query algorithm after making a fixed number of queries to a classifier, what is the expected value of each query to the adversary and what is the best query strategy for a fixed number of queries?*

The final challenge for real-world evasion is to design algorithms that can thwart attempts to evade the classifier. Promising potential defensive techniques include randomizing the classifier and identifying queries and sending misleading responses to the adversary. However, no proven defense against evasion has thus far been proposed.

5. PRIVACY VIOLATIONS

Privacy-preserving learning has been studied by research communities in Security, Databases, Theory, Machine Learning and Statistics. The broad goal of this research is to release aggregate statistics on a dataset without disclosing local information about individual data elements.[3] In the language of our taxonomy, privacy-preserving learning should be robust to Exploratory or Causative attacks which aim to violate Privacy. An attacker with access to a released statistic, model or classifier may probe it in an attempt to reveal information about the training data; moreover an attacker with influence over some proportion of the training examples may attempt to manipulate the mechanism into revealing information about unknown training examples. In this way the Privacy violation represents an important extension of the security violations of machine learning algorithms considered by our original taxonomy.

5.1 Differential Privacy

Historically, formal measures for quantifying the level of privacy preserved by a data analysis or data release have been elusive. Numerous definitions have been proposed and put aside due to the propositions being of a syntactic rather than semantic nature, most notably k-anonymity and its variants [61, 41]. However recently the concept of differential privacy due to Dwork [19] has emerged as a strong guarantee of privacy, with formal roots influenced by cryptography. This definition has enjoyed a significant amount of interest in the Theory community [17, 6, 22, 19, 1, 7, 24, 23, 44, 33, 25, 21, 3, 31, 59] where the general consensus is that the formal definition is meaningful and appropriately strong, while allowing for statistical learning methods that preserve the notion of privacy to be of practical use [56, 43, 1, 33, 17, 25, 3, 31, 11]. We now proceed to recall the definition of differential privacy and then to discuss its prevailing features in the current context of adversarial machine learning.

A *database* D is a sequence of *rows* $x_1, \ldots, x_n$ that are typically binary or real vectors but could belong to any domain $\mathcal{X}$. Given access to D, a *mechanism* M is tasked with releasing aggregate information about D while maintaining privacy of individual rows. In particular we assume that the *response* $M(D) \in \mathcal{T}_M$ is the only information released by the mechanism. This response could be a scalar statistic on D such as a mean, median or variance; or a model such as the parameters to a estimated joint density or the weight vectors to a learned classifier. We say that a pair of databases D_1, D_2 are *neighbors* if they differ on one row. With these definitions in-hand we can describe the following formal measure of privacy.

Definition 1 ([19]). *For any $\epsilon > 0$, a randomized mechanism M achieves* ϵ-differential privacy *if, for all pairs of neighboring databases D_1, D_2 and all responses $t \in \mathcal{T}_M$ the mechanism satisfies*[4]

$$\log\left(\frac{\Pr\left(M(D_1)=t\right)}{\Pr\left(M(D_2)=t\right)}\right) \quad \leq \quad \epsilon \ .$$

To understand this definition at a high level, consider a differentially private mechanism M that preserves data privacy by adding noise to the response of some desired non-private deterministic statistic $S(D)$, say the average $n^{-1}\sum_{i=1}^n x_i$ of a sequence of n scalars $x_1, \ldots, x_n$. The definition's likelihood ratio compares the distributions of M's noisy mean responses, when one scalar x_i (a database row) is changed. If the likelihood ratio is small (when privacy level $\epsilon \ll 1$), then the likelihood of M responding with noisy mean t on database D_1 is exceedingly close to the likelihood of responding with the same t on database D_2 with perturbed x_i: the mechanism's response distributions on the two *neighboring* databases are point-wise close.

5.1.1 Example: Private SVM Learning

As a more practical example we have previously studied differentially private mechanisms for support vector machine (SVM) learning [56]. There the setting is again a private database on which we wish to perform inference. However the database is now composed of rows of feature vectors and binary labels making up a training set of supervised binary classification. The desired inference is now the more sophisticated task of SVM learning [14]: in the linear case find a hyperplane normal vector that maximizes margin on the training set, and in the non-linear case perform this margin-maximization in a high-dimensional feature space induced by a user-defined kernel function.

The mechanism here responds with the weight vector representing the learned classifier itself; the response is the parameterization of a function. Our mechanism for linear SVM simply adds Laplace noise to the weight vector, which we prove using the algorithmic stability of SVM learning to achieve differential privacy. For the non-linear case we first solve linear SVM in a random feature space with inner-product approximating the desired kernel before adding noise to the corresponding solution; this first step allows us to achieve differential privacy even for kernels such as the Radial Basis Function that corresponds to learning in an infinite-dimensional feature space.

Another approach to differentially private SVM learning is due to Chaudhuri *et al.* [11] who instead of adding noise to the solution of SVM learning, randomly perturb the optimization used for SVM learning itself.

Numerous other practical learning algorithms have been made differentially private including regularized logistic regression [10], several collaborative filtering algorithms [43], point estimation [59], nearest neighbor, histograms, perceptron [6], and more.

5.2 Exploratory & Causative Privacy Attacks

An important observation on differential privacy is that the definition provides for very strong, semantic guarantees

[3] An orthogonal body of work exists, in which secure multi-party computation is applied to machine learning [18, 29]. There one has to share individual data points with untrusted 3$^{\text{rd}}$ parties in order to train a learner; *e.g.*, when employing cloud-based computation, or pooling with others' data to train a more predictive model. The resulting model or classifier, however, does not preserve training data privacy.

[4] The probabilities in the definition are over the randomization of mechanism M not over the databases which are fixed.

of privacy. Even with knowledge of M up to randomness and with knowledge of (say) the first $n-1$ rows D, an adversary cannot learn any additional information on the n^{th} row from a sublinear (in n) sample of $M(D)$. The adversary may even attempt a brute-force exploratory attack with such auxiliary information and unbounded computational resources:

1. For each possible x'_n consider $D' = x_1, \ldots, x_{n-1}, x'_n$ neighboring database D
   (a) Offline: Calculate the response distribution $p_{D'}$ of $M(D')$ by simulation.
2. Estimate the distribution of $M(D)$ as $\hat{p}_D$ by querying the mechanism repeatedly (a sublinear number of times).
3. Identify $x_n = x'_n$ by the $p_{D'}$ most closely resembling $\hat{p}_D$.

However for high levels of privacy (sufficiently small ϵ), the sampling error in $\hat{p}_D$ will be greater than the differences between alternate $p_{D'}$, and so even this powerful brute-force exploratory attack will fail with high probability. The same robustness holds even in the setting of the analogous causative attack, where the adversary can arbitrary manipulate the first $n-1$ rows.

5.3 On the Role of Randomization

As suggested above, the predominant avenue for designing a differentially private version of a statistic, model or learning algorithm is to produce a randomized mechanism that corresponds to the desired non-private algorithm, with noise added at some stage of the inference.

The most common approach to achieving differential privacy is to add noise to the target non-private mechanism's response $S(D)$, typically Laplace noise [22, 19, 6, 20, 56] but more generally an exponential mechanism is used to randomly select a response based on distance to $S(D)$ [44, 7, 33]. In the former case, the scale of Laplace noise is directly proportional to the sensitivity of the target mechanism S's response to changing between neighboring databases, and inversely proportional to ϵ. Often-times the sensitivity itself decreases with increasing database size, and so for larger databases less noise is added to guarantee the same level of differential privacy.

Differential privacy has also been achieved by randomizing the objective function of an optimization performed to execute learning. For example Chaudhuri *et al.* have developed differentially private versions of empirical risk minimizers including the SVM [11] and regularized logistic regression [10]. In their work, the original learning algorithm is formulated as an optimization typically minimizing the weighted sum of empirical risk and a regularization term. By adding an additional term which is the inner-product between the weight vector and a random direction, learning tends to slightly prefer the random direction and in so doing can be proven to yield differential privacy under certain technical conditions.

The role of randomization of the desired statistic or learning algorithm, either through adding output noise, randomizing an objective function, or similar, is crucial in providing a differentially private mechanism that can release aggregate information on a database while preserving the privacy of individual data. While we have suspected randomization could prove beneficial to fighting attacks that violate integrity or availability [51], few positive results are known.

5.4 Utility in the Face of Randomness

The more a target non-private estimator is randomized the more privacy is preserved, but at a cost to utility. Several researchers have considered this inherent trade-off between privacy and *utility*.

In our work on differentially private SVM learning [56], we define the utility of our private mechanism to be the pointwise difference between released privacy-preserving classifiers and non-private SVM classifiers. A private classifier (trained on D) that with high probability yields very similar classifications to an SVM (trained on D), for all test points, is judged to be of high utility since it well-approximates the desired non-private SVM classifier. Similar notions of utility are considered by Barak *et al.* [1] when releasing contingency tables whose marginals are close to true marginals; Blum *et al.* [7] whose mechanism releases anonymized data on which a class of analyses yield similar results as on the original data; and Kasiviswanathan *et al.* [33] and Beimel *et al.* [3] who consider utility as corresponding to PAC learning where response and target concepts learned on sensitive data are averaged over the underlying measure. Others such as Chaudhuri *et al.* [10, 11] measure the utility of a differential private mechanism not by its approximation of a target non-private algorithm, but rather by the absolute error it achieves. In all of these works, the differentially private mechanism is analyzed with the chosen utility in mind to produce an upper bound on the utility achieved by that particular mechanism.

Fundamental limits on the trade-off between differential privacy and utility have also been of great interest in past work, through negative results (lower bounds) that essentially state that mechanisms cannot achieve both high levels of privacy preservation and utility simultaneously. In our work on differentially private SVM learning [56] we establish lower bounds for approximating both linear and RBF SVM learning with any differentially private mechanism, quantifying levels of differential privacy and utility that cannot be achieved together. Dinur & Nissim [17] show that if noise of rate only $o(\sqrt{n})$ is added to subset sum queries on a database D of bits then an adversary can reconstruct a $1 - o(1)$ fraction of D: if accuracy is too great then privacy cannot be guaranteed at all. Hardt & Talwar [31] and Beimel *et al.* [3] conducted more recent studies establishing upper and lower bounds for the trade-off between utility and privacy in respective settings where the mechanism responds with linear transformations of data, and the setting of private PAC learning.

While significant progress has been made in achieving differential privacy and utility, understanding connections between differential privacy and learnability [3], algorithmic stability [56], robust statistics [21], and even mechanism design [44], many open problems remain in finding more complete understandings of these connections, making practical learning algorithms differentially private, and understanding the trade-off between privacy and utility.

6. CONCLUSION

The field of adversarial machine learning can be thought of as studying the effects of bringing the "Byzantine" to machine learning. In this paper, we showed how two machine learning methods, SpamBayes and PCA-based network anomaly detection, are vulnerable to causative attacks and discussed how application domain, features and data distribution restrict an adversary's actions.

We also outlined models for the adversary's capabilities, in terms of input corruption and class limitations, and their knowledge of the learning systems algorithm, feature space, and input data. We also considered how an adversary attacks a learning system, and discussed defenses and countermeasures. We examined exploratory attacks against learning systems and presented an important theoretical result in the field of near optimal evasion, showing that it was easy for adversaries to search convex classifiers to find input that can avoid being classified as negative.

Finally, we explored approaches and challenges for privacy-preserving learning, including differential privacy, exploratory and causative privacy attacks, and randomization.

7. ACKNOWLEDGMENTS

We would like to thank Marco Barreno, Peter Bartlett, Battista Biggio, Chris Cai, Fuching Jack Chi, Michael Jordan, Marius Kloft, Pavel Laskov, Shing-hon Lau, Steven Lee, Satish Rao, Udam Saini, Russell Sears, Charles Sutton, Nina Taft, Anthony Tran, and Kai Xia for many fruitful discussions and collaborations that have influenced our thinking about adversarial machine learning. We gratefully acknowledge the support of our sponsors. This work was supported in part by TRUST (Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation; DETERlab (cyber-DEfense Technology Experimental Research laboratory), which receives support from DHS HSARPA; and the Berkeley Counter-Censorship Incubator, which receives support from the US Department of State; and by the Alexander von Humboldt Foundation. We also received support from: Amazon, Cisco, Cloudera, eBay, facebook, Fujitsu, Google, HP, Intel, Microsoft, NetApp, Oracle, SAP, VMware, and Yahoo! Research. The opinions expressed in this paper are solely those of the authors and do not necessarily reflect the opinions of any sponsor.

8. REFERENCES

[1] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *PODS'07*, pages 273–282, 2007.

[2] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. Can machine learning be secure? In *ASIACCS'06*, pages 16–25, 2006.

[3] A. Beimel, S. Kasiviswanathan, and K. Nissim. Bounds on the sample complexity for private learning and private data release. In *Theory of Crypto.*, volume 5978 of *LNCS*, pages 437–454. 2010.

[4] B. Biggio, G. Fumera, and F. Roli. Multiple classifier systems under attack. In *Proc. Int. Workshop Multiple Classifier Systems*, volume 5997, pages 74–83, 2010.

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[6] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the SuLQ framework. In *PODS'05*, pages 128–138, 2005.

[7] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC'08*, pages 609–618, 2008.

[8] M. Brückner and T. Scheffer. Nash equilibria of static prediction games. In *NIPS*, pages 171–179. 2009.

[9] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[10] K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In *NIPS*, pages 289–296, 2009.

[11] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *JMLR*, 12:1069–1109, 2011.

[12] S. P. Chung and A. K. Mok. Allergy attack against automatic signature generation. In *RAID'09*, volume 4219 of *LNCS*, pages 61–80, 2006.

[13] S. P. Chung and A. K. Mok. Advanced allergy attacks: Does a corpus really help? In *RAID'07*, volume 4637 of *LNCS*, pages 236–255, 2007.

[14] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[15] C. Croux, P. Filzmoser, and M. R. Oliveira. Algorithms for projection-pursuit robust principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 87(2):218–225, 2007.

[16] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *KDD'04*, pages 99–108, 2004.

[17] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS'03*, pages 202–210, 2003.

[18] Y. Duan, J. Canny, and J. Zhan. P4P: Practical large-scale privacy-preserving distributed computation robust against malicious users. In *USENIX Security*, pages 207–222, 2010.

[19] C. Dwork. Differential privacy. In *ICALP'06*, pages 1–12, 2006.

[20] C. Dwork. A firm foundation for private data analysis. *Comms. ACM*, 54(1):86–95, 2011.

[21] C. Dwork and J. Lei. Differential privacy and robust statistics. In *STOC'09*, pages 371–380, 2009.

[22] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC'06*, pages 265–284, 2006.

[23] C. Dwork, F. McSherry, and K. Talwar. The price of privacy and the limits of LP decoding. In *STOC'07*, pages 85–94, 2007.

[24] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC'09*, pages 381–390, 2009.

[25] C. Dwork and S. Yekhanin. New efficient attacks on statistical disclosure control mechanisms. In *CRYPTO'08*, pages 469–480, 2008.

[26] R. A. Fisher. Question 14: Combining independent tests of significance. *American Statistician*, 2(5):30–31, 1948.

[27] P. Fogla and W. Lee. Evading network anomaly detection systems: Formal reasoning and practical techniques. In *CCS'06*, pages 59–68, 2006.

[28] A. Globerson and S. Roweis. Nightmare at test time: Robust learning by feature deletion. In *ICML'06*, pages 353–360, 2006.

[29] R. Hall, S. Fienberg, and Y. Nardi. Secure multiparty linear regression based on homomorphic encryption. *J. Official Statistics*, 2011. To appear.

[30] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. Probability and Mathematical Statistics. John Wiley and Sons, 1986.

[31] M. Hardt and K. Talwar. On the geometry of differential privacy. In *STOC'10*, pages 705–714, 2010.

[32] M. Kantarcioglu, B. Xi, and C. Clifton. Classifier evaluation and attribute selection against active adversaries. Technical Report 09-01, Purdue University, February 2009.

[33] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? In *FOCS'08*, pages 531–540, 2008.

[34] A. Kerckhoffs. La cryptographie militaire. *Journal des Sciences Militaires*, 9:5–83, January 1883.

[35] M. Kloft and P. Laskov. Online anomaly detection under adversarial impact. In *AISTATS'10*, 2010.

[36] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *SIGCOMM'04*, pages 219–230, 2004.

[37] P. Laskov and M. Kloft. A framework for quantitative security analysis of machine learning. In *AISec'09*, pages 1–4, 2009.

[38] C. Liu and S. Stamm. Fighting unicode-obfuscated spam. In *Proceedings of the Anti-Phishing Working Groups 2^{nd} Annual eCrime Researchers Summit*, pages 45–59, 2007.

[39] D. Lowd and C. Meek. Adversarial learning. In *KDD'05*, pages 641–647, 2005.

[40] D. Lowd and C. Meek. Good word attacks on statistical spam filters. In *CEAS'05*, 2005.

[41] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM Trans. KDD*, 1(1), 2007.

[42] M. V. Mahoney and P. K. Chan. An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. In *RAID'03*, volume 2820 of *LNCS*, pages 220–237, 2003.

[43] F. McSherry and I. Mironov. Differentially private recommender systems: building privacy into the net. In *KDD'09*, pages 627–636, 2009.

[44] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS'07*, pages 94–103, 2007.

[45] T. A. Meyer and B. Whateley. SpamBayes: Effective open-source, Bayesian based, email classification system. In *CEAS'04*, 2004.

[46] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[47] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia. Exploiting machine learning to subvert your spam filter. In *LEET'08*, pages 1–9, 2008.

[48] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia. Misleading learners: Co-opting your spam filter. In J. J. P. Tsai and P. S. Yu, editors, *Machine Learning in Cyber Trust: Security, Privacy, Reliability*, pages 17–51. Springer, 2009.

[49] B. Nelson and A. D. Joseph. Bounding an attack's complexity for a simple learning model. In *Proc. Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*, 2006.

[50] B. Nelson, B. I. P. Rubinstein, L. Huang, A. D. Joseph, S. hon Lau, S. Lee, S. Rao, A. Tran, and J. D. Tygar. Near-optimal evasion of convex-inducing classifiers. In *AISTATS*, 2010.

[51] B. Nelson, B. I. P. Rubinstein, L. Huang, A. D. Joseph, and J. D. Tygar. Classifier evasion: Models and open problems (position paper). In *Proc. Workshop on Privacy & Security issues in Data Mining and Machine Learning*, 2010.

[52] J. Newsome, B. Karp, and D. Song. Paragraph: Thwarting signature learning by training maliciously. In *RAID*, volume 4219 of *LNCS*, pages 81–105, 2006.

[53] L. Rademacher and N. Goyal. Learning convex bodies is hard. In *COLT*, pages 303–308, 2009.

[54] H. Ringberg, A. Soule, J. Rexford, and C. Diot. Sensitivity of PCA for traffic anomaly detection. In *SIGMETRICS*, pages 109–120, 2007.

[55] G. Robinson. A statistical approach to the spam problem. *Linux Journal*, Mar. 2003.

[56] B. I. P. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft. Learning in a large function space: Privacy-preserving mechanisms for SVM learning, 2009. In submission; `http://arxiv.org/abs/0911.5708v1`.

[57] B. I. P. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S. hon Lau, S. Rao, N. Taft, and J. D. Tygar. ANTIDOTE: Understanding and defending against poisoning of anomaly detectors. In A. Feldmann and L. Mathy, editors, *IMC'09*, pages 1–14, New York, NY, USA, November 2009. ACM.

[58] D. Sculley, G. M. Wachman, and C. E. Brodley. Spam filtering using inexact string matching in explicit feature space with on-line linear classifiers. In *TREC'06*, 2006.

[59] A. Smith. Privacy-preserving statistical estimation with optimal convergence rates. In *STOC'2011*, pages 813–822, 2011.

[60] S. J. Stolfo, W. jen Li, S. Hershkop, K. Wang, C. wei Hu, and O. Nimeskern. Detecting viral propagations using email behavior profiles. In *ACM Trans. Internet Technology*, May 2004.

[61] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[62] K. M. C. Tan, K. S. Killourhy, and R. A. Maxion. Undermining an anomaly-based intrusion detection system using common exploits. In *RAID'02*, volume 2516 of *LNCS*, pages 54–73, 2002.

[63] S. Venkataraman, A. Blum, and D. Song. Limits of learning-based signature generation with adversaries. In *NDSS'08*, 2008.

[64] D. Wagner and P. Soto. Mimicry attacks on host-based intrusion detection systems. In *CCS'02*, pages 255–264, 2002.

[65] G. L. Wittel and S. F. Wu. On attacking statistical spam filters. In *CEAS'04*, 2004.