The "Game" of Intrusion Detection

J. D. Tygar
University of California, Berkeley

Abstract: *We can view intrusion detection as a game, placing us firmly in the emerging field of adversarial machine learning. In adversarial machine learning, opponents deliberately attempt to generate data that causes traditional machine learning algorithms to behave poorly in security applications. This paper gives a brief overview of the field, and discusses several attacks and defenses, as well giving theoretical limits derived from a study of near-optimal evasion.*

Keywords: machine learning, adversarial machine learning, computer security, spam e-mail, intrusion detection

Computer security is often viewed as a game where various players make moves. An attacker (opponent) makes a move, and in response a defender makes a move, which in turn causes the opponent to make a new move, and so on. This powerful paradigm has long proven effective at helping us to analyze computer security. So let us apply the "game" paradigm to statistical machine learning in computer security applications. Advocates of machine learning argue, with good reason, that it is a powerful technique: if machines can learn when a system is functioning normally and when it is under attack, then we can build mechanisms that automatically and rapidly respond to emerging attacks. Such a system might be able to automatically screen out a wide variety of spam, phishing, network intrusions, malware, and other nasty Internet behavior. But the actual deployment of machine learning in computer security has been less successful than we might hope. What accounts for the difference?

Attacks on machine learning systems

To understand the issues, it is helpful to look "under the hood" more closely at what happens when we use machine learning. Perhaps the most popular model is *supervised learning*, in which we train a system using *labeled data* – that is data that has been marked as "attack" or "benign." For example, in a spam email detector, we would label a set of training email messages as *spam* or *ham* (although it doesn't sound very kosher, "ham" is a term used to denote non-spam email). The machine learning algorithm then produces a classifier, which takes unlabeled email messages as input, then classifies them as likely spam or ham. During training, a classifier is likely to learn that terms such as "Viagra" or "V1@gr@," for example, are a strong indicator of likely spam.

In this model the move of the opponent is to try to cause the supervised learning system to behave poorly. The opponent will try to craft input that causes it to misbehave.

Good machine learning algorithms are designed to perform well even if they get some random badly labeled input (such as a spam message that is accidentally mislabeled as ham). However, in the context of computer security, this does not go far enough. Adversaries (in this case, spammers) might play dirty by creating an adversarial training set: instead of sending "normal" spam, they might send (Byzantine) "tricky" spam designed to make the classifier misbehave.

This is not merely a theoretical observation. Here are some fragments from some apparent tricky spam email messages that my colleagues and I have collected (complete with original spelling and punctuation):

- "what, is he coming home, and without poor lydia?" she cried. "sure he will not leave London
- "i am quite sorry, lizzy, that you should be forced to have that disagreeable man all to yourself.
- calvert dawson blockage card. coercion choreograph asparagine bonnet contrast bloop. coextensive bodybuild bastion chalkboard denominate clare churchgo compote act. childhood ardent brethren commercial complain concerto depressor.
- brocade crown bethought chimney. angelo asphyxiate brad abase decompression codebreak. crankcase big conjuncture chit contention acorn cpa bladderwort chick. cinematic agleam chemisorb brothel choir conformance airfield.

Do you recognize any part of these messages? The first two fragments are quotes from Jane Austen's *Pride and Prejudice*. The second two messages are lists of less-common words in English. These tricky spam messages poison the training set. When they are labeled as spam and fed to a machine learning algorithm, they dilute the quality of spam detection. The algorithm could infer a rule that a benign term (such as "Lydia," "London," "brethren," or "chimney") is actually a marker for spam. When the classifier begins to label its inputs, it will generate false positives: ham that is incorrectly marked as spam. Large numbers of false positives undermine users' confidence in the learning algorithm. In practice, users find that their spam detectors seem tone-deaf and often misclassify email, requiring them to constantly check their "likely spam" mailboxes to manually retrieve misclassified ham.

Other types of attacks are also possible. For example, in systems that continually retrain, an adversary might try a "boiling-frog" attack. (Legend has it that if you drop a frog in a boiling pot of water, it will quickly jump out; but if you put a frog in lukewarm water and then slowly raise the heat, the frog cannot detect the slow change and will ultimately be boiled.) Consider using machine learning to detect abnormal network traffic. In a boiling-frog attack, an adversary slowly introduces aberrant input, and the system learns to tolerate it. Ultimately, the classifier learns to tolerate more and more aberrant input, until the adversary can launch a full-scale attack without detection.

These examples help to motivate the new science of *adversarial learning* – the development of machine learning algorithms that are effective even when adversaries play dirty. Adversaries have a variety of goals – we have identified at least three independent axes describing types of adversarial attacks:

- **Influence**: *Causative* (try to influence data and training) vs. *Exploratory* (probing during the test phase)
- **Security target:** *Integrity* (generate false negatives) vs. *Availability* (generate false positives)
- **Specificity:** *Targeted* (influence the classification of a particular input) vs. *Indiscriminate* (influence the classification of all types of inputs)

Hardening machine learning

These examples highlight the failings of classical machine learning. The good news is that a new science of adversarial machine learning is emerging — the development of algorithms that are effective even when adversaries play dirty.

My colleagues and I at UC Berkeley — as well as other research teams around the world — have been looking at these problems and developing new machine learning algorithms that are robust against adversarial input. One technique that we've used with great success is Reject On Negative Impact (RONI). In RONI, we screen training input to make sure that no single input substantially changes our classifier's behavior. This has a cost (we need a larger training set), but it also forces the adversary to control a much larger fraction of the input to mis-train the classifier.

The search for adversarial machine learning algorithms is thrilling: it combines the best work in robust statistics, machine learning, and computer security. One significant tool security researchers use is the ability to look at attack scenarios from the adversary's perspective (the *black hat* approach), and in that way, show the limits of computer security techniques. In the field of adversarial machine learning, this approach yields fundamental insights. Even though a growing number of adversarial machine learning algorithms are available, the black hat approach shows us that there are some theoretical limits to their effectiveness.

One powerful family of results that come from the black hat approach is called *near-optimal evasion*. We start by "thinking like a spammer." Suppose we want to sell Viagra via unsolicited email. If we try a direct approach, we're certain to have our email automatically classified as spam. So, we'll try to avoid this by modifying our message. For example, instead of using an email subject line such as "Cheap Online Pharmacy," we can try a subject line that promises instead a "Moderate Online Apothecary." We assume that we have sufficient access to a spam detector that we can pre-test our messages to see whether they're classified as spam. First, we identify our positive target spam message hawking Viagra. We cannot send this message because it is certain to be identified as spam. We call our target message "positive" because the classifier will give it a positive classification as spam. At the other end, we find some message that's completely benign and that avoids detection as spam. We call this our "negative" instance (because the classifier returns a negative result: it is not spam.) So now we have two extremes. We can perform a type of binary search — finding intermediate messages between these two extremes. When we get two messages that are close to each other — one classified as spam, the other classified as ham — we know we are near

the classifier's boundary. We can send the message that is classified as ham, and we say that it is "nearly optimal" but evades detection.

Now, we turn the tables again (switching roles in the "game") and resume the role of defender. We naturally ask: Can we stop this black hat attack? It turns out that for an important type of classifier, known as *convex classifiers*, we cannot stop it. The binary search strategy of a spammer is simply too strong. This shows the boundaries of the underlying theoretical limits of what is possible in adversarial machine learning. To get beyond them, we will either need to make our systems more complicated (going beyond convex classifiers) or use a fundamentally new strategy that no long depends as much on machine learning.

While some of the questions in this field have a theoretical flavor, at the end of the day, this is not a theoretical field. We need real-world machine learning algorithms that perform well even in adversarial environments. And while various research groups around the world are hard at work developing powerful adversarial machine learning algorithms, more work is needed before machine learning can fulfill its full promise in improving our cybersecurity algorithms. (To find out more about the field and the examples I mention, visit http://radlab.cs.berkeley.edu/wiki/SecML).

Acknowledgments: The work I mention is joint research with a number of researchers listed at http://radlab.cs.berkeley.edu/wiki/SecML. I would especially like to acknowledge my collaborators Marco Barreno, Anthony Joseph, Ling Huang, Blaine Nelson, Benjamin Rubinstein, and Satish Rao.