Outside the Closed World: On Using Machine Learning For Network Intrusion Detection

Robin Sommer
International Computer Science Institute, and
Lawrence Berkeley National Laboratory

Vern Paxson
International Computer Science Institute, and
University of California, Berkeley

***Abstract*—In network intrusion detection research, one popular strategy for finding attacks is monitoring a network's activity for *anomalies*: deviations from profiles of normality previously learned from benign traffic, typically identified using tools borrowed from the machine learning community. However, despite extensive academic research one finds a striking gap in terms of actual deployments of such systems: compared with other intrusion detection approaches, machine learning is rarely employed in operational "real world" settings. We examine the differences between the network intrusion detection problem and other areas where machine learning regularly finds much more success. Our main claim is that the task of finding attacks is fundamentally different from these other applications, making it significantly harder for the intrusion detection community to employ machine learning effectively. We support this claim by identifying challenges particular to network intrusion detection, and provide a set of guidelines meant to strengthen future research on anomaly detection.**

***Keywords*-anomaly detection; machine learning; intrusion detection; network security.**

I. Introduction

Traditionally, network intrusion detection systems (NIDS) are broadly classified based on the style of detection they are using: systems relying on *misuse-detection* monitor activity with precise descriptions of known malicious behavior, while *anomaly-detection* systems have a notion of normal activity and flag deviations from that profile.[1] Both approaches have been extensively studied by the research community for many years. However, in terms of actual deployments, we observe a striking imbalance: in operational settings, of these two main classes we find almost exclusively only misuse detectors in use—most commonly in the form of signature systems that scan network traffic for characteristic byte sequences.

This situation is somewhat striking when considering the success that machine-learning—which frequently forms the basis for anomaly-detection—sees in many other areas of computer science, where it often results in large-scale deployments in the commercial world. Examples from other domains include product recommendations systems such as used by Amazon [3] and Netflix [4]; optical character recognition systems (e.g., [5], [6]); natural language translation [7]; and also spam detection, as an example closer to home [8].

In this paper we set out to examine the differences between the intrusion detection domain and other areas where machine learning is used with more success. Our main claim is that the task of finding attacks is fundamentally different from other applications, making it significantly harder for the intrusion detection community to employ machine learning effectively. We believe that a significant part of the problem already originates in the premise, found in virtually any relevant textbook, that anomaly detection is suitable for finding *novel* attacks; we argue that this premise does not hold with the generality commonly implied. Rather, the strength of machine-learning tools is finding activity that is *similar to something previously seen*, without the need however to precisely describe that activity up front (as misuse detection must).

In addition, we identify further characteristics that our domain exhibits that are not well aligned with the requirements of machine-learning. These include: *(i)* a very high cost of errors; *(ii)* lack of training data; *(iii)* a semantic gap between results and their operational interpretation; *(iv)* enormous variability in input data; and *(v)* fundamental difficulties for conducting sound evaluation. While these challenges may not be surprising for those who have been working in the domain for some time, they can be easily lost on newcomers. To address them, we deem it crucial for any effective deployment to acquire deep, semantic *insight* into a system's capabilities and limitations, rather than treating the system as a black box as unfortunately often seen.

We stress that we do *not* consider machine-learning an inappropriate tool for intrusion detection. Its use requires care, however: the more crisply one can define the context in which it operates, the better promise the results may hold. Likewise, the better we understand the semantics of the detection process, the more operationally relevant the system will be. Consequently, we also present a set of guidelines meant to strengthen future intrusion detection research.

[1] Other styles include *specification-based* [1] and *behavioral* detection [2]. These approaches focus respectively on defining allowed types of activity in order to flag any other activity as forbidden, and analyzing patterns of activity and surrounding context to find secondary evidence of attacks.

Throughout the discussion, we frame our mindset around on the goal of using an anomaly detection system effectively in the "real world", i.e., in large-scale, operational environments. We focus on *network* intrusion detection as that is our main area of expertise, though we believe that similar arguments hold for host-based systems. For ease of exposition we will use the term *anomaly detection* somewhat narrowly to refer to detection approaches that rely primarily on machine-learning. By "machine-learning" we mean algorithms that are first trained with reference input to "learn" its specifics (either supervised or unsupervised), to then be deployed on previously unseen input for the actual detection process. While our terminology is deliberately a bit vague, we believe it captures what many in the field intuitively associate with the term "anomaly detection".

We structure the remainder of the paper as follows. In Section II, we begin with a brief discussion of machine learning as it has been applied to intrusion detection in the past. We then in Section III identify the specific challenges machine learning faces in our domain. In Section IV we present recommendations that we hope will help to strengthen future research, and we briefly summarize in Section V.

II. Machine Learning in Intrusion Detection

Anomaly detection systems find deviations from *expected behavior*. Based on a notion of *normal* activity, they report deviations from that profile as alerts. The basic assumption underlying any anomaly detection system—malicious activity exhibits characteristics not observed for normal usage—was first introduced by Denning in her seminal work on the host-based IDES system [9] in 1987. To capture normal activity, IDES (and its successor NIDES [10]) used a combination of statistical metrics and profiles. Since then, many more approaches have been pursued. Often, they borrow schemes from the machine learning community, such as information theory [11], neural networks [12], support vector machines [13], genetic algorithms [14], artificial immune-systems [15], and many more. In our discussion, we focus on anomaly detection systems that utilize such machine learning approaches.

Chandola et al. provide a survey of anomaly detection in [16], including other areas where similar approaches are used, such as monitoring credit card spending patterns for fraudulent activity. While in such applications one is also looking for outliers, the data tends to be much more structured. For example, the space for representing credit card transactions is of relatively low dimensionality and semantically much more well-defined than network traffic [17].

Anomaly detection approaches must grapple with a set of well-recognized problems [18]: the detectors tend to generate numerous false positives; attack-free data for training is hard to find; and attackers can evade detection by gradually teaching a system to accept malicious activity as benign. Our discussion in this paper aims to develop a different general point: that much of the difficulty with anomaly detection systems stems from using tools borrowed from the machine learning community in inappropriate ways.

Compared to the extensive body of research, anomaly detection has not obtained much traction in the "real world". Those systems found in operational deployment are most commonly based on statistical profiles of heavily aggregated traffic, such as Arbor's Peakflow [19] and Lanscope's StealthWatch [20]. While highly helpful, such devices operate with a much more specific focus than with the generality that research papers often envision.[2] We see this situation as suggestive that many anomaly detection systems from the academic world do not live up to the requirements of operational settings.

III. Challenges of Using Machine Learning

It can be surprising at first to realize that despite extensive academic research efforts on anomaly detection, the success of such systems in operational environments has been very limited. In other domains, the very same machine learning tools that form the basis of anomaly detection systems have proven to work with great success, and are regularly used in commercial settings where large quantities of data render manual inspection infeasible. We believe that this "success discrepancy" arises because the intrusion detection domain exhibits particular characteristics that make the effective deployment of machine learning approaches *fundamentally* harder than in many other contexts.

In the following we identify these differences, with an aim of raising the community's awareness of the unique challenges anomaly detection faces when operating on network traffic. We note that our examples from other domains are primarily for illustration, as there is of course a continuous spectrum for many of the properties discussed (e.g., spam detection faces a similarly adversarial environment as intrusion detection does). We also note that we are network security researchers, *not* experts on machine-learning, and thus we argue mostly at an intuitive level rather than attempting to frame our statements in the formalisms employed for machine learning. However, based on discussions with colleagues who work with machine learning on a daily basis, we believe these intuitive arguments match well with what a more formal analysis would yield.

A. Outlier Detection

Fundamentally, machine-learning algorithms excel much better at finding similarities than at identifying activity that does not belong there: the classic machine learning application is a *classification* problem, rather than discovering meaningful outliers as required by an anomaly detection system [21]. Consider product recommendation systems such as that used by Amazon [3]: it employs *collaborative*

[2]We note that for commercial solutions it is always hard to say what they do *exactly*, as specifics of their internals are rarely publicly available.

filtering, matching each of a user's purchased (or positively rated) items with other *similar* products, where similarity is determined by products that tend be bought together. If the system instead operated like an anomaly detection system, it would look for items that are typically *not* bought together—a different kind of question with a much less clear answer, as according to [3], many product pairs have no common customers.

In some sense, outlier detection is also a classification problem: there are two classes, "normal" and "not normal", and the objective is determining which of the two more likely matches an observation. However, a basic rule of machine-learning is that one needs to train a system with specimens of *all* classes, and, crucially, the number of representatives found in the training set for *each* class should be large [22]. Yet for anomaly detection aiming to find *novel* attacks, by definition one cannot train on the attacks of interest, but *only* on normal traffic, and thus having only one category to compare new activity against.

In other words, one often winds up training an anomaly detection system with the opposite of what it is supposed to find—a setting certainly not ideal, as it requires having a *perfect* model of normality for any reliable decision. If, on the other hand, one had a classification problem with multiple alternatives to choose from, then it would suffice to have a model just crisp enough to separate the classes. To quote from Witten et al. [21]: *The idea of specifying only positive examples and adopting a standing assumption that the rest are negative is called the* closed world assumption. *. . . [The assumption] is not of much practical use in real-life problems because they rarely involve "closed" worlds in which you can be certain that all cases are covered.*

Spam detection is an example from the security domain of successfully applying machine learning to a classification problem. Originally proposed by Graham [8], Bayesian frameworks trained with large corpora of both spam *and* ham have evolved into a standard tool for reliably identifying unsolicited mail.

The observation that machine learning works much better for such true classification problems then leads to the conclusion that anomaly detection is likely in fact better suited for finding *variations of known attacks*, rather than previously unknown malicious activity. In such settings, one can train the system with specimens of the attacks as they are known *and* with normal background traffic, and thus achieve a much more reliable decision process.

B. High Cost of Errors

In intrusion detection, the relative cost of any misclassification is extremely high compared to many other machine learning applications. A false positive requires spending expensive analyst time examining the reported incident only to eventually determine that it reflects benign underlying activity. As argued by Axelsson, even a very small rate of false positives can quickly render an NIDS unusable [23]. False negatives, on the other hand, have the potential to cause serious damage to an organization: even a single compromised system can seriously undermine the integrity of the IT infrastructure. It is illuminating to compare such high costs with the impact of misclassifications in other domains:

- *Product recommendation systems* can readily tolerate errors as these do not have a direct negative impact. While for the seller a good recommendation has the potential to increase sales, a bad choice rarely hurts beyond a lost opportunity to have made a more enticing recommendation. (In fact, one might imagine such systems *deliberately* making more unlikely guesses on occasion, with the hope of pointing customers to products they would not have otherwise considered.) If recommendations do not align well with the customers' interest, they will most likely just continue shopping, rather than take a damaging step such as switching to different seller. As Greg Linden said (author of the recommendation engine behind Amazon): "Recommendations involve a lot of guesswork. Our error rate will always be high." [24]
- *OCR technology* can likewise tolerate errors much more readily than an anomaly detection system. Spelling and grammar checkers are commonly employed to clean up results, weeding out the obvious mistakes. More generally, statistical language models associate probabilities with results, allowing for postprocessing of a system's initial output [25]. In addition, users have been trained not to expected perfect documents but to proofread where accuracy is important. While this corresponds to verifying NIDS alerts manually, it is much quicker for a human eye to check spelling of a word than to validate a report of, say, a web server compromise. Similar to OCR, contemporary automated language translation operates at relatively large errors rates [7], and while recent progress has been impressive, nobody would expect more than a rough translation.
- *Spam detection* faces a highly unbalanced cost model: false positives (i.e., ham declared as spam) can prove very expensive, but false negatives (spam not identified as such) do not have a significant impact. This discrepancy can allow for "lopsided" tuning, leading to systems that emphasize finding obvious spam fairly reliably, yet exhibiting less reliability for new variations hitherto unseen. For an anomaly detection system that primarily aims to find novel attacks, such performance on new variations rarely constitutes an appropriate trade-off.

Overall, an anomaly detection system faces a much more stringent limit on the number of errors that it can tolerate.

However, the intrusion detection-specific challenges that we discuss here all tend to *increase* error rates—even above the levels for other domains. We deem this unfortunate combination as the primary reason for the lack of success in operational settings.

C. Semantic Gap

Anomaly detection systems face a key challenge of transferring their results into *actionable* reports for the network operator. In many studies, we observe a lack of this crucial final step, which we term the *semantic gap*. Unfortunately, in the intrusion detection community we find a tendency to limit the evaluation of anomaly detection systems to an assessment of a system's capability to reliably *identify* deviations from the normal profile. While doing so indeed comprises an important ingredient for a sound study, the next step then needs to interpret the results from an operator's point of view—"What does it *mean*?"

Answering this question goes to the heart of the difference between finding "abnormal activity" and "attacks". Those familiar with anomaly detection are usually the first to acknowledge that such systems are *not* targeting to identify malicious behavior but just report what has not been seen before, whether benign or not. We argue however that one cannot stop at that point. After all, the objective of deploying an intrusion detection system *is* to find attacks, and thus a detector that does not allow for bridging this gap is unlikely to meet operational expectations. The common experience with anomaly detection systems producing too many false positives supports this view: by definition, a machine learning algorithm does not make any mistakes *within* its model of normality; yet for the operator it is the results' interpretation that matters.

When addressing the semantic gap, one consideration is the incorporation of local security policies. While often neglected in academic research, a fundamental observation about operational networks is the degree to which they *differ*: many security constraints are a site-specific property. Activity that is fine in an academic setting can be banned in an enterprise network; and even inside a single organization, department policies can differ widely. Thus, it is crucial for a NIDS to accommodate such differences.

For an anomaly detection system, the natural strategy to address site-specifics is having the system "learn" them during training with normal traffic. However, one cannot simply assert this as the solution to the question of adapting to different sites; one needs to explicitly *demonstrate* it, since the core issue concerns that such variations can prove diverse and easy to overlook.

Unfortunately, more often than not security policies are not defined crisply on a technical level. For example, an environment might tolerate peer-to-peer traffic as long as it is not used for distributing inappropriate content, and that it remains "below the radar" in terms of volume. To report a violation of such a policy, the anomaly detection system would need to have a notion of what is deemed "appropriate" or "egregiously large" *in that particular environment*; a decision out of reach for any of today's systems. Reporting just the usage of P2P applications is likely not particularly useful, unless the environment flat-out bans such usage. In our experience, such vague guidelines are actually common in many environments, and sometimes originate in the imprecise legal language found in the "terms of service" to which users must agree [26].

The basic challenge with regard to the semantic gap is understanding how the *features* the anomaly detection system operates on relate to the semantics of the network environment. In particular, for any given choice of features there will be a fundamental limit to the kind of determinations a NIDS can develop from them. Returning to the P2P example, when examining only NetFlow records, it is hard to imagine how one might spot inappropriate content.[3] As another example, consider exfiltration of personally identifying information (PII). In many threat models, loss of PII ranks quite high, as it has the potential for causing major damage (either directly, in financial terms, or due to publicity or political fallout). On a technical level, some forms of PII are not that hard to describe; e.g., social security numbers as well bank account numbers follow specific schemes that one can verify automatically.[4] But an anomaly detection system developed in the absence of such descriptions has little hope of finding PII, and even given examples of PII and non-PII will likely have difficulty distilling rules for accurately distinguishing one from the other.

D. Diversity of Network Traffic

Network traffic often exhibits much more diversity than people intuitively expect, which leads to misconceptions about what anomaly detection technology can realistically achieve in operational environments. Even within a single network, the network's most basic characteristics—such as bandwidth, duration of connections, and application mix—can exhibit immense variability, rendering them unpredictable over short time intervals (seconds to hours). The

[3] We note that in fact the literature holds some fairly amazing demonstrations of how much more information a dataset can provide than what we might intuitively expect: Wright et al. [27] infer the language spoken on *encrypted* VOIP sessions; Yen et al. [28] identify the particular web browser a client uses from flow-level data; Narayanan et al. [29] identify users in the anonymized Netflix datasets via correlation with their public reviews in a separate database; and Kumar et al. [30] determine from lossy and remote packet traces the number of disks attached to systems infected by the "Witty" worm, as well as their uptime to millisecond precision. However these examples all demonstrate the power of exploiting *structural* knowledge informed by very careful examination of the particular *domain* of study—results not obtainable by simply expecting an anomaly detection system to develop inferences about "peculiar" activity.

[4] With limitations of course. As it turns out, Japanese phone numbers look a lot like US social security numbers, as the Lawrence Berkeley National Laboratory noticed when monitoring for them in email [31].

widespread prevalence of strong correlations and "heavy-tailed" data transfers [32], [33] regularly leads to large bursts of activity. It is crucial to acknowledge that in networking such variability occurs regularly; it does not represent anything unusual. For an anomaly detection system, however, such variability can prove hard to deal with, as it makes it difficult to find a stable notion of "normality".

One way to reduce the diversity of Internet traffic is to employ *aggregation*. While highly variable over small-to-medium time intervals, traffic properties tend to greater stability when observed over longer time periods (hours to days, sometimes weeks). For example, in most networks time-of-day and day-of-week effects exhibit reliable patterns: if during today's lunch break, the traffic volume is twice as large as during the corresponding time slots last week, that likely reflects something unusual occurring. Not coincidentally, one form of anomaly detection system we *do* find in operation deployment is those that operate on highly aggregated information, such as "volume per hour" or "connections per source". On the other hand, incidents found by these systems tend to be rather noisy anyway—and often straight-forward to find with other approaches (e.g., simple threshold schemes). This last observation goes to the heart of what can often undermine anomaly detection research efforts: a failure to examine whether simpler, non-machine learning approaches might work equally well.

Finally, we note that traffic diversity is not restricted to packet-level features, but extends to application-layer information as well, both in terms of syntactic and semantic variability. Syntactically, protocol specifications often purposefully leave room for interpretation, and in heterogeneous traffic streams there is ample opportunity for corner-case situations to manifest (see the discussion of "crud" in [34]). Semantically, features derived from application protocols can be just as fluctuating as network-layer packets (see, e.g., [35], [36]).

E. Difficulties with Evaluation

For an anomaly detection system, a thorough evaluation is particularly crucial to perform, as experience shows that many promising approaches turn out in practice to fall short of one's expectations. That said, devising sound evaluation schemes is not easy, and in fact turns out to be *more difficult than building the detector itself*. Due to the opacity of the detection process, the results of an anomaly detection system are harder to predict than for a misuse detector. We discuss evaluation challenges in terms of the difficulties for *(i)* finding the right data, and then *(ii)* interpreting results.

1) Difficulties of Data: Arguably the most significant challenge an evaluation faces is the lack of appropriate public datasets for assessing anomaly detection systems. In other domains, we often find either standardized test suites available, or the possibility to collect an appropriate corpus, or both. For example, for automatic language translation "a large training set of the input-output behavior that we seek to automate is available to us *in the wild*" [37]. For spam detectors, dedicated "spam feeds" [38] provide large collections of spam free of privacy concerns. Getting suitable collections of "ham" is more difficult, however even a small number of private mail archives can already yield a large corpus [39]. For OCR, sophisticated methods have been devised to generate ground-truth automatically [40]. In our domain, however, we often have neither standardized test sets, nor any appropriate, readily available data.

The two publicly available datasets that have provided something of a standardized setting in the past—the DARPA/Lincoln Labs packet traces [41], [42] and the KDD Cup dataset derived from them [43]—are now a decade old, and no longer adequate for any current study. The DARPA dataset contains multiple weeks of network activity from a simulated Air Force network, generated in 1998 and refined in 1999. Not only is this data synthetic, and no longer even close to reflecting contemporary attacks, but it also has been so extensively studied over the years that most members of the intrusion detection community deem it wholly uninteresting if a NIDS now reliably detects the attacks it contains. (Indeed, the DARPA data faced pointed criticisms not long after its release [44], particularly regarding the degree to which simulated data can be appropriate for the evaluation of a NIDS.) The KDD dataset represents a distillation of the DARPA traces into features for machine learning. Not only does it inherit the shortcomings of the DARPA data, but the features have also turned out to exhibit unfortunate artifacts [45].

Given the lack of publicly available data, it is natural to ask why we find such a striking gap in our community.[5] The primary reason clearly arises from the data's sensitive nature: the inspection of network traffic can reveal highly sensitive information, including confidential or personal communications, an organization's business secrets, or its users' network access patterns. Any breach of such information can prove catastrophic not only for the organization itself, but also for affected third parties. It is understandable that in the face of such high risks, researchers frequently encounter insurmountable organizational and legal barriers when they attempt to provide datasets to the community.

Given this difficulty, researchers have pursued two alternative routes in the past: simulation and anonymization. As demonstrated by the DARPA dataset, network traffic generated by simulation can have the major benefit of being free of sensitivity concerns. However, Internet traffic

[5]We note that the lack of public network data is not limited to the intrusion detection domain. We see effects similar to the overuse of the DARPA dataset in empirical network research: the *ClarkNet-HTTP* [46] dataset contains two weeks' worth of HTTP requests to ClarkNet's web server, recorded in 1995. While researchers at ClarkNet stopped using these logs for their own studies in 1997, in total researchers have used the traces for evaluations in more than 90 papers published between 1995 and 2007—13 of these in 2007 [47]!

is already exceedingly difficult to simulate realistically by itself [48]. Evaluating an anomaly detection system that strives to find novel attacks using only simulated activity will often lack any plausible degree of realism or relevance.

One can also sanitize captured data by, e.g., removing or anonymizing potentially sensitive information [49], [50], [51]. However, despite intensive efforts [52], [53], publishing such datasets has garnered little traction to date, mostly one suspects for the fear that information can still leak. (As [54] demonstrates, this fear is well justified.) Furthermore, even if a scrubbed dataset is available, its use with an anomaly detection system can be quite problematic, since by definition such systems look precisely for the kind of artifacts that tend to be removed during the anonymization process [55].

Due to the lack of public data, researchers are forced to assemble their own datasets. However, in general this is *not* an easy task, as most lack access to appropriately sized networks. It is crucial to realize that activity found in a small laboratory network differs *fundamentally* from the aggregate traffic seen upstream where NIDSs are commonly deployed [26]. Conclusions drawn from analyzing a small environment cannot be generalized to settings of larger scale.

There is unfortunately no general answer to countering the lack of data for evaluation purposes. For any study it is thus crucial to *(i)* acknowledge shortcomings that one's evaluation dataset might impose, and *(ii)* consider alternatives specific to the particular setting. We return to these points in Section IV-D1.

2) Mind the Gap: The semantic gap requires any study to perform an explicit final step that tends to be implicit in other domains: changing perspective to that of a user of the system. In addition to correctly *identifying* attacks, an anomaly detection system also needs to support the operator in understanding the activity and enabling a quick assessment of its impact. Suppose a system correctly finds a previously unknown web server exploit, yet only reports it as "HTTP traffic of host did not match the normal profile". The operator will spend significant additional effort figuring out what happened, even if already having sufficient trust in the system to take its alerts seriously. In other applications of machine learning, we do not see a comparable problem, as results tend to be intuitive there. Returning to spam detection again, if the detector reports a mail as spam, there is not much room for interpretation left.

We argue that when evaluating an anomaly detection system, understanding the system's semantic properties—the *operationally relevant* activity that it can detect, as well as the *blind spots* every system will necessarily have—is much more valuable than identifying a concrete set of parameters for which the system happens to work best for a particular input. The specifics of network environments differ too widely to allow for predicting performance in other settings based on just numbers. Yet, with insight into the *conceptual capabilities* of a system, a network operator can judge a detector's potential to support different operational concerns as required. That said, we note that Tan et al. demonstrated the amount of effort it can require to understand a single parameter's impact, even with an conceptually simple anomaly detection system [56] .

3) Adversarial Setting: A final characteristic unique to the intrusion detection domain concerns the adversarial environment such systems operate in. In contrast, users of OCR systems won't try to conceal characters in the input, nor will Amazon customers have much incentive (or opportunity) to mislead the company's recommendation system. Network intrusion detection, however, must grapple with a classic arms-race: attackers and defenders each improve their tools in response to the other side devising new techniques. One particular, serious concern in this regard is *evasion*: attackers adjusting their activity to avoid detection. While evasion poses a fundamentally hard problem for any NIDS [57], anomaly detection faces further risks due to the nature of underlying machine learning. In [58], Fogla and Lee present an automated approach to mutate attacks so that they match a system's normal profile. More generally, in [59] Barreno et al. present a taxonomy of attacks on machine-learning systems.

From a research perspective, addressing evasion is a stimulating topic to explore; on *theoretical* grounds it is what separates intrusion detection most clearly from other domains. However, we argue that from a *practical* perspective, the impact of the adversarial setting is not necessarily as significant as one might initially believe. Exploiting the specifics of a machine learning implementation requires significant effort, time, and expertise on the attacker's side. Considering that most of today's attacks are however *not* deliberately targeting handpicked victims—yet simply exploit whatever sites they find vulnerable, indiscriminantly seeking targets— the risk of an anomaly detector falling victim to a sophisticated evasion attack is small in many environments. Assuming such a threat model, it appears prudent to focus first on addressing the many other challenges in using machine learning effectively, as they affect a system's operational performance more severely.

IV. Recommendations for Using Machine Learning

In light of the points developed above, we now formulate guidelines that we hope will help to strengthen future research on anomaly detection. We note that we view these guidelines as touchstones rather than as firm rules; there is certainly room for further discussion within the wider intrusion detection community.

If we could give only *one* recommendation on how to improve the state of anomaly detection research, it would be: *Understand what the system is doing*. The intrusion detection community does not benefit any further from yet another

study measuring the performance of some previously untried combination of a machine learning scheme with a particular feature set, applied to something like the DARPA dataset. The nature of our domain is such that one can always find a variation that works slightly better than anything else in a particular setting. Unfortunately, while obvious for those working in the domain for some time, this fact can be easily lost on newcomers. Intuitively, when achieving better results on the same data than anybody else, one would expect this to be a definite contribution to the progress of the field. The point we wish to convey however is that we are working in an area where *insight* matters much more than just numerical results.

A. Understanding the Threat Model

Before starting to develop an anomaly detector, one needs to consider the anticipated threat model, as that establishes the framework for choosing trade-offs. Questions to address include:

- *What kind of environment does the system target?* Operation in a small network faces very different challenges than for a large enterprise or backbone network; academic environments impose different requirements than commercial enterprises.
- *What do missed attacks cost?* Possible answers ranges from "very little" to "lethal." A site's determination will depend on its security demands as well as on other deployed attack detectors.
- *What skills and resources will attackers have?* If a site deems itself at high risk for explicit *targeting* by an attacker, it needs to anticipate much more sophisticated attacks than those incurred by potential victims of indiscriminant "background radiation" activity.
- *What concern does evasion pose?* The degree to which attackers might analyze defense techniques and seek to circumvent them determines the robustness requirements for any detector.

There are no perfect detectors in intrusion detection—hence one always must settle for less-than-ideal solutions. However, operators can make informed decisions only when a system's threat model is clearly stated.

B. Keeping The Scope Narrow

It is crucial to have a clear picture of what problem a system targets: what specifically are the attacks to be detected? The more narrowly one can define the target activity, the better one can tailor a detector to its specifics and reduce the potential for misclassifications.

Of course machine-learning is not a "silver bullet" guaranteed to appropriately match a particular detection task. Thus, *after* identifying the activity to report, the next step is a neutral assessment of what constitutes the right sort of tool for the task; in some cases it will be an anomaly detector, but in others a rule-based approach might hold more promise. A common pitfall is *starting* with the premise to use machine-learning (or, worse, a *particular* machine-learning approach) and then looking for a problem to solve. We argue that such a starting point is biased and thus rarely leads to the best solution to a problem.

When settling on a specific machine-learning algorithm as the appropriate tool, one should have an answer for *why* the particular choice promises to perform well in the intended setting—not only on strictly mathematical grounds, but considering domain-specific properties. As discussed by Duda et al. [22], there are "no *context-independent* [...] reasons to favor one learning [...] method over another" (emphasis added); they call this the "no free lunch theorem". Note that if existing systems target similar activity, it can be illuminating to understand their shortcomings to motivate how the proposed approach avoids similar problems.

A substantive part of answering the *Why?* question is identifying the feature set the detector will work with: insight into the features' significance (in terms of the domain) and capabilities (in terms of revealing the targeted activity) goes a long way towards reliable detection. A common pitfall here is the temptation to base the feature set on a dataset that happens to be at hand for evaluation. However, if one cannot make a solid argument for the relation of the features to the attacks of interest, the resulting study risks foundering on serious flaws.

A good example for the kind of mindset we deem vital for sound anomaly detection studies is the work on web-based attacks by Kruegel et al. [60]. From the outset, the authors focus on a very specific class of attacks: exploiting web servers with malformed query parameters. The discussion convincingly argues for the need of anomaly detection (such attacks share conceptual similarities, yet differ in their specifics sufficiently to make writing signatures impractical); and the authors clearly motivate the choice of features by comparing characteristics of benign and malicious requests (e.g., the typical length of a query's parameters tends to be short, while a successful buffer overflow attempt likely requires long shellcode sequences and padding). Laying out the land like this sets up the stage for a well-grounded study.

C. Reducing the Costs

Per the discussion in Section III-B, it follows that one obtains enormous benefit from reducing the costs associated with using an anomaly detection system. Anecdotally, the number one complaint about anomaly detection systems is the excessive number of false positives they commonly report. As we have seen, an anomaly detection system does not necessarily make *more* mistakes than machine learning systems deployed in other domains—yet the high cost associated with each error often conflicts with effective

operation. Thus, limiting false positives must be a top priority for any anomaly detection system.

Likely the most important step towards fewer mistakes is reducing the system's scope, as discussed in Section IV-B. Arguably, without a clear objective no anomaly detection system can achieve a tolerable amount of false positives without unacceptably compromising on its detection rate. The setup of the underlying machine-learning problem also has a direct impact on the number of false positives. Per Section III-A, machine-learning works best when trained using activity similar to that targeted for detection.

An anomaly detection system also requires a strategy to deal with the natural diversity of network traffic (Section III-D). Often, aggregating or averaging features over suitable time intervals proves helpful, assuming the threat model allows for coarser granularity. Another approach is to carefully examine the features for their particular properties; some will be more invariant than others. As a simple flow-level example, the set of destination ports a particular internal host contacts will likely fluctuate quite a bit for typical client systems; but we might often find the set of ports on which it *accepts* incoming connections to be stable over extended periods of time.

Finally, we can reduce false positives by post-processing them with the support of additional information. For example, Gu et al.'s "BotHunter" system uses a "statistical payload anomaly detection engine" as one tool among others (Snort signatures, and a typical scan detector), and a final stage correlates the output of all of them [61]. Likewise, Anagnostakis et al.'s "Shadow Honeypots" validate the results of anomaly detectors with an instrumented copy of the protected system [62]. If we find automated post-processing infeasible, we might still be able to reduce costs by providing the analyst with additional *information* designed to accelerate the manual inspection process.

D. Evaluation

When evaluating an anomaly detection system, the primary objective should be to develop *insight* into the system's capabilities: *What can it detect, and why? What can it not detect, and why not? How reliably does it operate? Where does it break?* In our experience, the #1 reason that conference submissions on anomaly detection fail arises from a failure to adequately explore these issues. We discuss evaluation separately in terms of working with data, and interpreting results.

1) Working with data: The single most important step for sound evaluation concerns obtaining appropriate data to work with. The "gold standard" here is obtaining access to a dataset containing *real network traffic* from as large an environment as possible[6]; and ideally multiple of these from different networks. Work with actual traffic greatly strengthens a study, as the evaluation can then demonstrate how well the system should work in practice. In our experience, the best way to obtain such data is to provide a clear benefit in return to the network's operators; either, ideally, by research that aims to directly help to improve operations, or by exchanging the access for work on an unrelated area of importance to the operators.

Note that the options for obtaining data differ with the setting, and it often pays to consider potential data sources *early* on when designing the detector. For example, honeypots [63] can provide data (usually) free of sensitivity concerns, though they cannot provide insight into how malicious traffic manifests differently from benign "background" traffic; or when working with companies that control large quantities of the data of interest, one might need to plan strategically by sending a student or staff member for an extended stay. Alternatively, *mediated* trace access can be a viable strategy [64]: rather than bringing the data to the experimenter, bring the experiment to the data, i.e., researchers send their analysis programs to data providers who then run them on their behalf and return the output.

Once acquired, the datasets require a careful assessment of their characteristics. To interpret results correctly, one must not only understand what the data contains, but also how it is *flawed*. No dataset is perfect: often measurements include artifacts that can impact the results (such as filtering or unintended loss), or unrelated noise that one can safely filter out *if* readily identified (e.g., an internal vulnerability scan run by the security department). See [65] for further discussion of issues relating to working with network data.

When evaluating an anomaly detection system, one *always* needs multiple datasets. First, one must train the system with different data than used for ultimate evaluation. (This is a basic requirement for sound science, yet overlooked surprisingly often; see however [21] for a set of standard techniques one can apply when having only limited data available). Perhaps less obviously, to demonstrate that the system can adapt to different environments through learning *requires* evaluation using data from multiple sources. We stress that, as noted in Section III-E1, the DARPA and KDD Cup traces cannot serve as viable datasets. Their only role in contemporary research is for basic functionality tests and cross-checking results (i.e., to test whether an approach is hopelessly broken).

Subdividing is a standard approach for performing training and detection on different traffic even when one has only a single dataset from the examined environment. It works by selecting subsets of the available data via random sampling.

[6]Results from large environments usually transfer directly to smaller networks, with the benefit that one can choose trade-offs more conservatively in the latter. However, results collected in small environments rarely apply directly to large ones.

Subdividing can work well if it is performed *in advance* of the actual study. Note however that the splitting must be unbiased with regards to the features the anomaly detection system examines. For example, when operating on a per-flow basis, one should flow-sample the dataset rather than packet-sample.

2) Understanding results: The most important aspect of interpreting results is to *understand their origins*. A sound evaluation frequently requires relating input and output on a very low-level. Researchers need to *manually examine* false positives. If when doing so one cannot determine why the system incorrectly reported a particular instance, this indicates a *lack of insight* into the anomaly detection system's operation. Note, one needs to relate such false positives to the *semantics* of the traffic; it is hardly helpful to frame them in the mathematical terms of the detection logic ("activity exceeded the distance metric's threshold"). If faced with too many false positives to manually examine, then one can employ random sampling to select an appropriately sized subset for direct inspection.

False negatives often prove harder to investigate than false positives because they require reliable *ground-truth*, which can be notoriously hard to obtain for an anomaly detection system that aims to spot previously unseen activity. Nevertheless, such an assessment forms a crucial part of the story and merits careful attention. It can be highly beneficial to consider the question of ground-truth already at the *beginning* of a study. If one cannot find a sound way to obtain ground-truth for the evaluation, then it becomes *questionable to pursue the work at all*, even if it otherwise appears on a solid foundation. One must collect ground-truth via a mechanism orthogonal (unrelated) to how the detector works. One approach is to use a different mechanism to label the input, with the obvious disadvantage that such input will only be as good as this other technique. (Sometimes a *subset* of the data can arguably be labeled in this fashion with high accuracy. If so, then provided that the subset is formed in a fashion independent from how the detector under development functions, one can extrapolate from performance on the subset to broader performance.) Another solution is manual labeling—often however infeasible given the large amount of data a NIDS operates on. A final compromise is to *inject* a set of attacks deemed representative of the kind the anomaly detection system should detect.

An important but often overlooked additional consideration is to include in an evaluation inspection of the *true* positives and negatives as well. This need arises from the opacity of the decision process: with machine learning, it is often not apparent *what* the system learned even when it produces correct results. A classic illustration of this problem comes from a Pentagon project from 1980s [66]: a neural network was trained to detect tanks in photos, and in the initial evaluation it was indeed able to correctly separate photos depicting tanks from those which did not. It turned out, however, that the datasets used for training and evaluation shared a subtle property: photos of tanks were taken on a cloudy day, while all others had a blue sky. As later cross-checking revealed, the neural network had simply learned to detect the color of the sky.

We can in fact turn around the notion of understanding the origins of anomaly detection results, changing the emphasis from gaining insight into how an anomaly detection system achieves its results to instead *illuminating the problem space*. That is, machine learning is often underappreciated as potentially providing a means to an end, rather than an end itself: one employs it not to ultimately detect malicious activity, but rather to understand the significance of the different features of benign and malicious activity, which then eventually serve as the basis for a *non-machine-learning* detector.

For example, consider spam classification. By examining which phrases a Bayesian classifier employs most effectively one might discover that certain parts of messages (e.g., subject lines, `Received` headers, MIME tags) provide disproportionate detection power. In this contrived example, one might then realize that a detector that *directly* examines those components—perhaps not employing any sort of Bayesian-based analysis, but instead building on separate domain knowledge—can provide more effective classification by leveraging the *structural* properties of the domain. Thus, machine learning can sometimes serve very effectively to "point the way" to how to develop detectors that are themselves based on different principles. (The idea here is similar to that employed in Principle Component Analysis, which aims to find which among a wide set of features contribute the most to particular clusters of activity [22].)

We note that such an approach can also help overcome potential performance bottlenecks. Many machine learning algorithms are best suited for offline batch operation, and less so for settings requiring low-latency real-time detection. Non-machine-learning detectors often prove significantly easier to implement in a streaming fashion even at high data rates.

A separate consideration concerns how an evaluation compares results with other systems found in the literature. Doing so requires care to ensure fair treatment. The successful operation of an anomaly detection system typically requires significant experience with the particular system, as it needs to be tuned to the local setting—experience that can prove cumbersome to collect if the underlying objective is instead to understand the *new* system. Nevertheless, as a first step a comparative study needs to *reproduce* the results reported in the literature for the "foreign" system.

Finally, the most convincing real-world test of any anomaly detection system is to solicit feedback from operators who run the system in their network. If they genuinely deem the system helpful in their daily routine, that provides compelling support for the study.

V. Conclusion

Our work examines the surprising imbalance between the extensive amount of research on machine learning-based anomaly detection pursued in the academic intrusion detection community, versus the lack of operational deployments of such systems. We argue that this discrepancy stems in large part from specifics of the problem domain that make it significantly harder to apply machine learning effectively than in many other areas of computer science where such schemes are used with greater success. The domain-specific challenges include: *(i)* the need for *outlier detection*, while machine learning instead performs better at finding *similarities*; *(ii)* very high costs of classification errors, which render error rates as encountered in other domains unrealistic; *(iii)* a semantic gap between detection results and their operational interpretation; *(iv)* the enormous variability of benign traffic, making it difficult to find stable notions of normality; *(v)* significant challenges with performing sound evaluation; and *(vi)* the need to operate in an adversarial setting. While none of these render machine learning an inappropriate tool for intrusion detection, we deem their unfortunate combination in this domain as a primary reason for its lack of success.

To overcome these challenges, we provide a set of guidelines for applying machine learning to network intrusion detection. In particular, we argue for the importance of obtaining *insight* into the operation of an anomaly detection system in terms of its capabilities and limitations *from an operational point of view*. It is crucial to acknowledge that the nature of the domain is such that one can *always* find schemes that yield marginally better ROC curves than anything else has for a specific given setting. Such results however do not contribute to the progress of the field without any *semantic* understanding of the gain.

We hope for this discussion to contribute to strengthening future research on anomaly detection by pinpointing the fundamental challenges it faces. We stress that we do not consider our discussion as final, and we look forward to the intrusion detection community engaging in an ongoing dialog on this topic.

Acknowledgments

We would like to thank Gerald Friedland for discussions and feedback, as well as the anonymous reviewers for their valuable suggestions. This work was supported in part by NSF Awards NSF-0433702 and CNS-0905631. Opinions, findings, and conclusions or recommendations are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work was also supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

[1] C. Ko, M. Ruschitzka, and K. Levitt, "Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification-based Approach," in *Proc. IEEE Symposium on Security and Privacy*, 1997.

[2] D. R. Ellis, J. G. Aiken, K. S. Attwood, and S. D. Tenaglia, "A Behavioral Approach to Worm Detection," in *Proc. ACM CCS WORM Workshop*, 2004.

[3] G. Linden, B. Smith, and J. York, "Amazon.com Recommendations: Item-to-Item Collaborative Filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[4] J. Bennett, S. Lanning, and N. Netflix, "The Netflix Prize," in *Proc. KDD Cup and Workshop*, 2007.

[5] L. Vincent, "Google Book Search: Document Understanding on a Massive Scale," 2007.

[6] R. Smith, "An Overview of the Tesseract OCR Engine," in *Proc. International Conference on Document Analysis and Recognition*, 2007.

[7] F. J. Och and H. Ney, "The Alignment Template Approach to Statistical Machine Translation," *Comput. Linguist.*, vol. 30, no. 4, pp. 417–449, 2004.

[8] P. Graham, "A Plan for Spam," in *Hackers & Painters*. O'Reilly, 2004.

[9] D. E. Denning, "An Intrusion-Detection Model," *IEEE Transactions on Software Engineering*, vol. 13, no. 2, pp. 222–232, 1987.

[10] H. S. Javitz and A. Valdes, "The NIDES Statistical Component: Description and Justification," SRI International, Tech. Rep., 1993.

[11] W. Lee and D. Xiang, "Information-Theoretic Measures for Anomaly Detection," in *Proc. IEEE Symposium on Security and Privacy*, 2001.

[12] Z. Zhang, J. Li, C. Manikopoulos, J. Jorgenson, and J. Ucles, "HIDE: a Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification," in *Proc. IEEE Workshop on Information Assurance and Security*, 2001.

[13] W. Hu, Y. Liao, and V. R. Vemuri, "Robust Anomaly Detection Using Support Vector Machines," in *Proc. International Conference on Machine Learning*, 2003.

[14] C. Sinclair, L. Pierce, and S. Matzner, "An Application of Machine Learning to Network Intrusion Detection," in *Proc. Computer Security Applications Conference*, 1999.

[15] S. A. Hofmeyr, "An Immunological Model of Distributed Detection and its Application to Computer Security," Ph.D. dissertation, University of New Mexico, 1999.

[16] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," University of Minnesota, Tech. Rep., 2007.

[17] R. J. Bolton and D. J. Hand, "Statistical Fraud Detection: A Review," *Statistical Science*, vol. 17, no. 3, 2002.

[18] C. Gates and C. Taylor, "Challenging the Anomaly Detection Paradigm: A Provocative Discussion," in *Proc. Workshop on New Security Paradigms*, 2007.

[19] "Peakflow SP," http://www.arbornetworks.com/en/peakflow-sp.html.

[20] "StealthWatch," http://www.lancope.com/products/.

[21] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques (2nd edition)*. Morgan Kaufmann, 2005.

[22] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd edition)*. Wiley Interscience, 2001.

[23] S. Axelsson, "The Base-Rate Fallacy and Its Implications for the Difficulty of Intrusion Detection," in *Proc. ACM Conference on Computer and Communications Security*, 1999.

[24] "Make Data Useful," Greg Linden, Data Mining Seminar, Stanford University, 2006. http://glinden.blogspot.com/2006/12/slides-from-my-talk-at-stanford.htm%l.

[25] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A General and Efficient Weighted Finite-state Transducer Library," in *Proc. International Conference on Implementation and Application of Automata*, 2007.

[26] R. Sommer, "Viable Network Intrusion Detection in High-Performance Environments," Ph.D. dissertation, TU München, 2005.

[27] C. V. Wright, L. Ballard, F. Monrose, and G. M. Masson, "Language Identification of Encrypted VoIP traffic: Alejandra y Roberto or Alice and Bob?" in *Proc. USENIX Security Symposium*, 2007.

[28] T.-F. Yen, X. Huang, F. Monrose, and M. K. Reiter, "Browser Fingerprinting from Coarse Traffic Summaries: Techniques and Implications," in *Proc. Conference on Detection of Intrusions and Malware & Vulnerability Assessment*, 2009.

[29] A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets," in *IEEE Symposium on Security and Privacy*, 2008.

[30] A. Kumar, V. Paxson, and N. Weaver, "Exploiting Underlying Structure for Detailed Reconstruction of an Internet-scale Event," in *ACM SIGCOMM Internet Measurement Conference*, 2005.

[31] Jim Mellander, Lawrence Berkeley National Laboratory, via personal communication, 2009.

[32] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, "Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level," *IEEE/ACM Transactions on Networking*, vol. 5, no. 1, 1997.

[33] A. Feldmann, A. C. Gilbert, and W. Willinger, "Data Networks As Cascades: Investigating the Multifractal Nature of Internet WAN Traffic," in *Proc. ACM SIGCOMM*, 1998.

[34] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," *Computer Networks*, vol. 31, no. 23–24, pp. 2435–2463, 1999.

[35] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "YouTube Traffic Characterization: A View From the Edge," in *Proc. ACM SIGCOMM Internet Measurement Conference*, 2008.

[36] A. Nazir, S. Raza, and C.-N. Chuah, "Unveiling Facebook: A Measurement Study of Social Network Based Applications," in *Proc. SIGCOMM Internet Measurement Conference*, 2008.

[37] A. Haley, P. Norvig, and F. Pereira, "The Unreasonable Effectiveness of Data," *IEEE Intelligent Systems*, March/April 2009.

[38] D. S. Anderson, C. Fleizach, S. Savage, and G. M. Voelker, "Spamscatter: Characterizing Internet Scam Hosting Infrastructure," in *Proc. USENIX Security Symposium*, 2007.

[39] G. V. Cormack and T. R. Lynam, "Online Supervised Spam Filter Evaluation," *ACM Transactions on Information Systems*, vol. 25, no. 3, 2007.

[40] J. van Beusekom, F. Shafait, and T. M. Breuel, "Automated OCR Ground Truth Generation," in *Proc. DAS 2008*, Sep 2008.

[41] R. Lippmann, R. K. Cunningham, D. J. Fried, I. Graf, K. R. Kendall, S. E. Webster, and M. A. Zissman, "Results of the 1998 DARPA Offline Intrusion Detection Evaluation," in *Proc. Recent Advances in Intrusion Detection*, 1999.

[42] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA Off-line Intrusion Detection Evaluation," *Computer Networks*, vol. 34, no. 4, pp. 579–595, October 2000.

[43] "KDD Cup Data," http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[44] J. McHugh, "Testing Intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratories," *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 262–294, November 2000.

[45] M. V. Mahoney and P. K. Chan, "An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection," in *Proc. Recent Advances in Intrusion Detection*, 2003.

[46] "ClarkNet-HTTP," http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html.

[47] Martin Arlitt, via personal communication, 2008.

[48] S. Floyd and V. Paxson, "Difficulties in Simulating the Internet," *IEEE/ACM Transactions on Networking*, vol. 9, no. 4, 2001.

[49] "tcpdpriv," http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html.

[50] J. Xu, J. Fan, M. Ammar, and S. Moon, "On the Design and Performance of Prefix-Preserving IP Traffic Trace Anonymization," in *Proc. ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001.

[51] R. Pang, M. Allman, V. Paxson, and J. Lee, "The Devil and Packet Trace Anonymization," in *Computer Communication Review*, 2006.

[52] "The Internet Traffic Archive (ITA)," http://ita.ee.lbl.gov.

[53] "PREDICT," http://www.predict.org.

[54] S. E. Coull, C. V. Wright, F. Monrose, M. P. Collins, and M. K. Reiter, "Playing Devil's Advocate: Inferring Sensitive Information from Anonymized Network Traces," in *Proc. Network and Distributed Security Symposium*, 2007.

[55] K. S. Killourhy and R. A. Maxion, "Toward Realistic and Artifact-Free Insider-Threat Data," *Proc. Computer Security Applications Conference*, 2007.

[56] K. M. Tan and R. A. Maxion, ""Why 6?" Defining the Operational Limits of Stide, an Anomaly-Based Intrusion Detector," in *Proc. IEEE Symposium on Security and Privacy*, 2002.

[57] T. H. Ptacek and T. N. Newsham, "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection," Secure Networks, Inc., Tech. Rep., January 1998.

[58] P. Fogla and W. Lee, "Evading Network Anomaly Detection Systems: Formal Reasoning and Practical Techniques," in *Proc. ACM Conference on Computer and Communications Security*, 2006.

[59] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can Machine Learning Be Secure?" in *Proc. ACM Symposium on Information, Computer and Communications Security*, 2006.

[60] C. Kruegel and G. Vigna, "Anomaly Detection of Web-based Attacks," in *Proc. ACM Conference on Computer and Communications Security*, 2003.

[61] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation," in *Proc. USENIX Security Symposium*, August 2007.

[62] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and A. D. Keromytis, "Detecting Targeted Attacks Using Shadow Honeypots," in *Proc. USENIX Security Symposium*, 2005.

[63] N. Provos and T. Holz, *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison Wesley, 2007.

[64] P. Mittal, V. Paxson, R. Sommer, and M. Winterrowd, "Securing Mediated Trace Access Using Black-box Permutation Analysis," in *Proc. ACM Workshop on Hot Topics in Networks*, 2009.

[65] V. Paxson, "Strategies for Sound Internet Measurement," in *ACM SIGCOMM Internet Measurement Conference*, Oct. 2004.

[66] N. Fraser, "Neural Network Follies," http://neil.fraser.name/writing/tank, 1998.