

Bridging the Semantic Gap: Human Factors in Anomaly-Based Intrusion Detection Systems

Richard Harang

1 Introduction

Network intrusion detection systems (IDSs) are often broadly divided into two broad groups; those that attempt to detect “misuse” (typically including signature-based approaches), and those that attempt to detect “anomalies” [1, 2]. Misuse detection typically relies on static rules (a.k.a. “signatures”) to detect malicious activity [3, 4]; activity that precisely matches a clearly defined set of rules is flagged as potentially malicious and referred for further action/analysis, while any activity that does not match any signature is assumed safe. This approach has so far proven quite effective, and the misuse detector Snort is perhaps one of the most widely deployed IDSs in use today. The misuse detection approach generally (although not always, see, e.g., [5]) benefits from an acceptably low false positive rate, and assuming adequate computational resources, is capable of detecting any attack it has a signature for. This low false positive rate comes at a price, however; due to the highly specific nature of the signatures that misuse detection requires, it is typically ineffective in detecting novel “zero-day” attacks targeting new vulnerabilities, for which no effective signature has yet been developed [6] and requires constant maintenance of the signature database obfuscated variants are developed (although see [7] for convincing arguments that anomaly-based detection will likely fare no better).

Anomaly-based detection generally attempts to fill a complementary role by applying the powerful and (in other domains) highly successful tools of machine learning and statistical analysis to the intrusion detection problem in order to detect variants on existing attacks or entirely new classes of attacks [8]. Such methods have become fairly widespread in the IDS academic literature, however very few anomaly detection systems appear to have been deployed to actual use [1], and none appear to be adopted on anything near the scale of misuse systems such as Snort [3] or Bro [4]. While a brief literature review will reveal hundred of papers examining methods

R. Harang (✉)

ICF International, Washington, DC, USA

e-mail: Richard.Harang@ICFI.com

ranging from self-organizing maps [9] to random forests [10] to active learning methods [11], there are very few that discuss operational deployment or long-term use of any of these approaches. Several potential reasons for this disparity are presented in [1], including their unacceptably high false positive rates (see also discussion in [12]), the severely skewed nature of the classification problem that IDS research presents, the lack of useful testing data, and the regrettable tendency of researchers in anomaly detection to treat the IDS as a black box with little regard to its operational role in the overall IDS process or the site-specific nature of security policies (referred to by [1] as the “semantic gap”).

In the following, we begin by examining the techniques used in several anomaly IDS implementations in the literature, and provide some discussion of them in terms of the “semantic gap” of Sommers and Paxson. We then provide a sketch of the base rate fallacy argument as it applies to intrusion detection, originally presented by Axelsson in [12]. We then extend the results of [12] briefly to show that the utility of an anomaly IDS is in fact closely related to the semantic gap as well as the false positive rate. Finally, we return to tree-based classifiers, present anecdotal evidence regarding their interpretability, and discuss methods for extracting rules from randomly constructed ensembles that appear to permit anomaly detection methods to greatly facilitate human interpretation of their results at only modest costs in accuracy.

2 Anomaly IDS Implementations

The field of anomaly detection has seen rapid growth in the past two decades, and a complete and exhaustive inventory of all techniques is prohibitive. We attempt to summarize the most popular implementations and approaches here with illustrative examples, rather than provide a comprehensive review.

Anomaly detection itself may be split into the two general categories of supervised and unsupervised learning [2, 8]. Supervised learning remains close to misuse detection, using a “training set” of known malicious and benign traffic in order to identify key predictors of malicious activity that will allow the detector to learn more general patterns that may detect both previously seen attacks as well as novel attacks not represented in the original training set. Unsupervised learning often relies on outlier detection approaches, reasoning that most use of a given system should be legitimate, and data points that appear to stand out from the rest of the data are more likely to represent potentially hostile activities.

2.1 *Categorical Outlier Detection*

In [13], a fairly general method for outlier detection in tabular categorical data is presented in which—for each column in each row—the relative frequency of the

entry is compared to all other entries in the given column; this method does not consider covariance between columns, and so has obvious links to e.g., Naïve Bayes models, but can be operated in $O(nm)$ time where n is the number of rows (data points) in the table, and m the number of columns (attributes). This method is most appropriate for purely categorical data where no metric or ordering exists, however they demonstrate that discretization of numerical data allows application of their method.

In [14], a slightly more general method is presented to allow for outlier detection on mixed categorical and numerical data, while permitting a varying number of categorical attributes per item, by examining set overlap for categorical attributes while monitoring conditional covariance relationships between the numerical attributes and performing weighted distance calculations between them. They also apply their method to the KDD'99 intrusion detection data set to examine its performance, reporting false positive rates of 3.5 %, with widely varying true positive rates per class of attack (from 95 to 0 %).

In the strictly continuous domain, the work of [15] examines conditional anomaly detection, i.e., detecting anomalous entries based on their context, by means of estimation from strictly parametric models. They fit a Gaussian mixture model to their observed data via maximum likelihood using the expectation–maximization algorithm, and detect outliers on the basis of low likelihood score. The work of [15] draws a rather clear distinction between “environment” variables that are conditioned on, and “indicator” variables that are examined for anomalies, however this might be readily extended by regenerating their models in sequence, once for each variable.

This class of outlier detection method generally forms one of the more interpretable ones; because individual points are generally considered in their original coordinate system (if one exists, c.f. support vector machines, below) or in terms of marginal probabilities, the explanation for why a particular point was classified as anomalous is generally amenable to introspection. For instance, in the work of [13], each field of an ‘anomalous’ entry may be inspected for its contribution to the score, and the ones contributing most to the score may be readily identified. The work of [14] is slightly more complex, due to the individualized covariance scoring, however the notion of set overlap is straightforward and the covariance relationship between variables lends itself to straightforward explanation. Unfortunately, despite their explanatory power, these methods generally do not appear to provide comparable performance to more complex methods such as those described below.

2.2 Support Vector Machines

Both supervised and unsupervised methods using support vector machines (SVMs) [16] have become extremely widespread due to the good generalization performance and high flexibility they afford. One-class SVMs are used in [17] to detect

“masquerades”, illicit use of a legitimate account by a third party posing as the legitimate user, and found to compare favorably to simpler techniques such as Naïve Bayes. One-class SVMs are also used in conjunction with conventional signature based approaches in [18] to provide enhanced detection of variations on existing signatures.

Supervised learning with two-class SVMs has also been applied to the KDD’99 data set in [19], and found to compare favorably to neural network techniques with respect to training time, however as the most widely used SVMs are by construction binary classifiers (multi-class SVMs are typically constructed on a 1-vs-all basis from multiple binary SVMs; SVM constructions that explicitly allow multiple classes exist but are computationally expensive [20]).

SVMs are also commonly used as a component of more complex systems; these hybrid approaches allow for the generalization power of SVMs to be exploited, while mitigating some of their common weaknesses, such as poor performance when dealing with ancillary statistics. In [21], the authors employ rough sets to reduce the feature space presented to SVMs for training and testing. A feature extraction step is also used to reduce the input space to one-class SVMs in [6], allowing them to better control the false positive rate.¹ Finally, [22] use a genetic algorithm to do initial feature extraction for a SVM, also using KDD’99 data; they also attempted to apply their method to real data, however concluded that their own network data contained too few attacks to reliably evaluate their method.

While strictly speaking, SVMs are simply maximum margin linear classifiers, SVMs as commonly used and understood derive their excellent classification properties from the “Kernel trick”, which uses the fact that data can be projected into certain inner product spaces (reproducing kernel Hilbert spaces) in which inner products may be computed through a kernel function on the original data without having to project the data into the space. These spaces are typically significantly more complex than the original space, but the kernel trick allows any classification method that may be written entirely in terms of inner products (such as support vectors) to be applied to those complex spaces based entirely on the original data, thus finding a separating hyperplane in the transformed space without ever explicitly constructing it.

While this projection into one of a wide range of high dimensional spaces gives SVMs their power, it also poses significant barriers to interpretability. Visualization and interpretation of (for instance) a 41-dimensional separating hyperplane is well beyond the capabilities of most people; developing an intuition for the effects of projection into what may be in principle an infinite dimensional space, and then considering the construction of a separating hyperplane in this space complicates matters substantially. For illustrative purposes, we have combined the approach of [23] (see below) with one-class support vector machines, and applied

¹ It is also worth noting that [6] use and provide access to a “KDD-like” set of data—that is, data aggregated on flow-like structures containing labels—that contains real data gathered from honeypots between 2006 and 2009 rather than synthetic attacks that predate 1999; this may provide a more useful and realistic alternative to the KDD’99 set.

```

####[ IP ]###
    version    = 4L
    ihl        = 5L
    tos        = 0x8
    len        = 1500
    id         = 47564
    flags      = DF
    frag       = 0L
    ttl        = 63
    proto      = tcp
    chksum     = 0x921d
    src        = 10.2.20.40
    dst        = 10.2.193.254

####[ TCP ]###
    sport      = 52711
    dport      = neod1
    seq        = 137580262
    ack        = 1222080868
    dataofs    = 5L
    reserved   = 0L
    flags      = A
    window     = 5840
    chksum     = 0xb0c1
    urgptr     = 0
    options    = []

####[ Raw ]###
0000  34 6E DD E9 76 BB 9F CD  19 8E 2A 46 95 87 72 38
0010  AD 47 BD 61 0C F5 B4 26  38 C7 6B 31 6B F5 6C FC
0020  98 2B 98 39 F1 56 61 F7  19 D5 B1 06 58 B4 D7 C4
0030  B8 DD D9 F0 0E 81 82 B1  7C 3A DE 53 EE 80 BB 27
0040  8E 73 2E 81 C5 C4 1D 22  E3 CE 07 5F D8 51 CF 88
...et cetera

```

Fig. 1 A packet within the support of the 1-class SVM

it to packet capture data from the West Point/NSA Cyber Defense Exercise [24]. Briefly, the histogram of 3-byte n-grams from each packet was constructed as a feature vector and provided to the 1-class SVM function within Scikit-Learn [25]; default values were used, and built-in functions were used to recover on support vector data point (Fig. 3). The norm imposed by the inner product derived from the kernel function was used to find the nearest point contained within the support of the SVM (Fig. 1), as well as the nearest point outside the support of the SVM (Fig. 2).² These figures provide Scapy [26] dissections of the three packets.

While the mathematical reason that the packet in Fig. 1 is considered ‘normal’ while that of Fig. 2 would be considered ‘anomalous’ is straightforward, from the

² A possibly instructive exercise for the reader: obscure the labels and ask a knowledgeable colleague to attempt to divine which two packets are ‘normal’ and which is ‘anomalous’, and why.

```

###[ IP ]###
  version   = 4L
  ihl       = 5L
  tos       = 0x8
  len       = 1500
  id        = 47565
  flags     = DF
  frag      = 0L
  ttl       = 63
  proto     = tcp
  chksum    = 0x921c
  src       = 10.2.20.40
  dst       = 10.2.193.254

###[ TCP ]###
  sport      = 52711
  dport      = neod1
  seq        = 137581722
  ack        = 1222080868
  dataofs    = 5L
  reserved   = 0L
  flags      = A
  window     = 5840
  chksum     = 0xcc4a
  urgptr     = 0
  options    = []

###[ Raw ]###
0000  C0 1C 65 35 A2 B9 44 C3  E0 30 40 BC E8 11 23 66
0010  A8 FF 6C 98 B6 82 10 C8  3A 99 DA 1E 25 44 B3 10
0020  62 07 F8 3C DD 15 FB 50  84 A2 04 20 F7 9B 0F E5
0030  C6 FA 61 EE FB 07 4B 22  0E 0A 6A 4E 24 B1 F1 2A
0040  6D 8F 85 80 E5 C3 6B 03  19 62 C2 B9 59 CD 45 91
...and so on

```

Fig. 2 A packet outside the support of the 1-class SVM

point of view of a human analyst, dealing with this output is difficult. In particular the transformed space of the SVM does not map neatly to the normal space of parameters that human analysts work within to determine hostile intent in network traffic, and the region of high-dimensional space, even assuming that analysts are willing and able to visualize point clouds in infinite dimensional space, does not neatly map to threat classes, misconfigurations, or other common causes of undesired activity. In effect, to either confirm or refute the classification of this anomaly detector, analysts must launch an entirely independent investigation into the packet at issue, with very little useful information on where the most fruitful areas for examination are likely to be.

```

###[ IP ]###
version    = 4L
ihl        = 5L
tos        = 0x8
len        = 1500
id         = 28176
flags      = DF
frag       = 0L
ttl        = 63
proto      = tcp
chksum     = 0xdddd9
src        = 10.2.20.40
dst        = 10.2.193.254

###[ TCP ]###
sport      = 32900
dport      = afrog
seq        = 83970169
ack        = 2996538417
dataofs    = 5L
reserved   = 0L
flags      = A
window     = 5840
chksum     = 0x3a3b
urgptr     = 0
options    = []

###[ Raw ]###
0000  5C 73 37 78 61 81 9F 17  27 29 A2 58 6C 83 5A CC
0010  F8 3B F5 47 8B 9A B3 81  DF 63 D7 82 84 9A AF 9A
0020  24 9B 7B D7 2E 5A D4 00  2A 37 E0 56 45 B1 4D 65
0030  7E 22 9E A5 F1 D8 6E 34  1B 8A 7C CF 36 E5 5F F4
0040  AD 2D 17 64 16 10 1A 33  E5 C1 8A 39 29 FF 06 78
0050  55 3C D8 4A 73 CB 28 E2  6D 8E 29 DA 2A 5C 08 B5

```

Fig. 3 The support vector

2.3 Clustering and Density Estimation

One of the more common forms of unsupervised learning is clustering. Such approaches in untransformed coordinates have also been examined; the work of [27] presents an application of an ensemble approach in which k-means clustering is used to on a per-port basis to form representative clusters for the data; new data is then assigned to a cluster which may accept or reject it as anomalous. In-place incremental updating of the clusters and their associated classification rules is used, and while the approach appears to underperform when compared to other, more complex ones applied to the KDD'99 data with a false positive rate of roughly 10 %, the method of cluster assignment and subsequent determination provides a degree of introspection to the decision not available to the more complex approaches.

The work of [23] addresses the problem of transforming the highly variable and often difficult to model content of individual network packets into a continuous space amenable to distance-based outlier detection via frequency counts of n -grams. For a given length n , all possible consecutive n -byte sequences are extracted from a packet and tabulated into a frequency table, which is then used to define a typical distribution for a given service (implicitly assuming a high-dimensional multivariate normal). Once this distribution has stabilized, further packets are then examined for consistency with this distribution, and ones that differ significantly are identified as anomalous. While the reported performance of the classifier is excellent (nearly 100 % accuracy on the DARPA IDS data set that formed the basis for the KDD'99 data), once packets are grouped per-port and per-service, the question of interpretability once again becomes a difficult one. While a priori known attacks in the data, including novel ones, were detected by their method, they do not discuss the issue of determining the true/false positive status of an alarm in unlabeled data in any detail.

2.4 Hybrid Techniques

The work of [11] represents an interesting case, which uses a combination of simulated data and active learning to convert the outlier detection problem into a more standard classification problem. Simulated “outlier” data is constructed either from a uniform distribution across a bounded subspace, or from the product distribution of the marginal distributions of the data, assuming independence, and adjoined to the original data with a label indicating its synthetic nature (see also [28], which uses random independent permutations of the columns to perform a similar task). The real and simulated data, labeled as such, is then passed to an ensemble classifier that attempts to maximize the margin between the two classes, using sampling to select points from regions where the margin is small to refine the separation between the classes. This method is also applied to KDD'99 data, where the results are actually shown to outperform the winning supervised method from the original contest. In discussing the motivation for their method, they point out that a “notable issue with [most outlier detection methods] ... is the lack of explanation for outlier flagging decisions.” While semantic issues are not addressed in further detail, the explicit classification-based approach they propose generates a reference distribution of ‘outliers’ that can be used as an aid to understand the performance of their outlier detection method.

Finally, the work of [9] combines both supervised and unsupervised learning approaches in order to attempt to leverage the advantages of each. They implement a C4.5 decision tree with labeled data to detect misuse, while using a self-organizing map (SOM) to detect anomalous traffic. Each new data point is presented to both techniques, and the output of both classifiers is considered in constructing the final classification of the point. While in this case the supervised learning portion of the detector could provide some level of semantic interpretation of anomalies,

should both detectors fire simultaneously, the question of how to deal with alerts that are triggered only by the SOM portion of the detector is not addressed.

2.5 *Tree-Based Classifiers*

Tree-based ensemble classifiers present alternatives to kernel-based approaches that rely upon projections of the data into higher-dimensional spaces. From their initial unified presentation in [28] on through the present day [29], they have been shown to have excellent performance on a variety of tasks. While their original applications focused primarily on supervised learning, various methods to adapt them to unsupervised approaches such as density estimation and clustering have been developed since their introduction (the interested reader is referred to [29] and references therein for an overview of the variety of ways that random decision forests have been adapted to various problems, as well as an excellent selection of references). Unsurprisingly, random decision forests and variants thereof have been applied to the anomaly IDS problem as well.

A straightforward application of random decision forests in a supervised learning framework using the KDD'99 data is provided in [30], reporting a classification accuracy rate of 99.8 %. Similar work in [10] using feature selection rather than feature importance yields similar results, reporting classification accuracy of 99.9 %. Our own experiments have shown that simply blindly applying the built-in random decision forest package in Scikit-learn [25] to the first three features of the KDD'99 data in alphabetical order—"Count", "diff_srv_rate", and "dst_bytes"—without any attempt to clean, balance, or otherwise adapt to the deficiencies of the KDD'99 set, yields over a 98 % accuracy rate in classifying KDD'99 traffic, with the bulk of the errors formed by misclassifications within the class of attacks (see Appendix for details).

An outlier detection approach for intrusion detection using a variant of random decision forests that the authors term "isolation forest" is presented in [31], in which the data is sub-sampled in extremely small batches (the authors recommend 256 points per isolation tree, and building a forest from 100 such trees), and iteratively split at random until the leaf size is reduced to some critical threshold. Using the intuition that anomalies should stand out from the data, it then follows that the fewer splits that are required to isolate a given point, the less similar to the rest of the data it is. Explicit anomaly scoring in terms of the expected path length of a binary search tree is computed. The authors of [31] examine—among other data sets—the portion of the KDD'99 data representing SMTP transactions, and report an AUC of 1.0 for the unsupervised detection of anomalies. Several of the same authors also present on-line versions of their algorithm in [32], taking advantage of the constant time complexity and extremely small memory footprint of their method to use sequential segments of data as training and test sets (where the test set from the previous iteration becomes the training set for the next iteration), and report similarly impressive results.

While the authors of [31] and [32] do not explicitly consider the notion of interpretation in their work, the earlier work of [33] provides an early example of an arguably similar approach that does explicitly consider such factors. Their anomaly detection method is explicitly categorical (with what amounts to a density estimation clustering approach to discretize continuous fields), and operates by generating a “rule forest” where frequently occurring patterns in commands and operations on host systems are used to generate several rule trees, which are then pruned based on quality of the rule and the number of observations represented by the rule.

Note, however, that in contrast to [31]—which also relies on trees—the approach of [33] lends more weight to longer rules that traverse more levels of the tree (where [31] searches for points that on average traverse very few levels of a tree before appearing in a leaf node). This occurs as the work of [31] (and outlier detection in general) tacitly assumes that the outlying data is not generated by the same process as the ‘inlying’ data, and hence the work of [31] searches for regions containing few points. Rule-based systems, such as [33] and [27] (and to a lesser extent, [34]) place more emphasis on conditional relationships in the data, in effect estimating the density for the final partitioning of the tree conditional on the prior path through the tree.

Also in contrast to many other anomaly IDS methods, [33] made interpretation of the results of their system—named “Wisdom and Sense”—an explicit design goal, noting that “anomaly resolution must be accomplished by a human.” The rules themselves, represented by the conjunction of categorical variables, lend themselves much more naturally to human interpretation than e.g., the high-dimensional projections common to support vector machines, or the laborious transformation of categorical data into vectors of binary data which is then blindly projected into the corners of a high-dimensional unit cube.

3 Anomaly IDSs “in the wild” and the Semantic Gap

Likely due to the complexities enumerated above, experimental results on attempts to deploy anomaly detection systems are regrettably few and far between. Earlier methods, such as that of [33] or that of [34], typically report results only on their own in-house and usually proprietary data. When these systems have been tested elsewhere (see [35], in which [34] was tested and extended), many configuration details have been found to be highly site-specific, and additional tuning was required. As the volume and sensitivity of network traffic increased, along with the computational complexity of the anomaly detection methods used to construct anomaly detectors, attempts to deploy such detectors to operational environments do not appear to be widely reported. The work of [36] provides a notable exception to this rule, in which several proprietary anomaly detection systems were deployed in a backbone environment. Details about the anomaly detection methods used in the commercial products tested in [36] are unavailable (and some are noted as also

incorporating ‘signature based methods’, though it is unclear at what level these signatures are being applied), however it is worth remarking that the bulk of the anomalies being examined in [36] were related to abnormal traffic patterns, rather than content (e.g., shell code, worms, Trojans, etc.). The volume of traffic and the extremely low rate of true positives meant that searching for more sophisticated attacks was essentially infeasible, and indeed the traffic patterns observed at the backbone level that were flagged as anomalous are much more straightforward to classify manually than more subtle exploitation attempts. Revisiting our terminology above, the cost α of evaluating a positive result and determining if it is a true or false positive is relatively low, at least in comparison to attacks that require content inspection to detect.

It is also worth remarking that in [36] it was observed that the set of true positive results on which the various anomaly detectors agreed was in fact extremely small, which suggests that the false negative rate for a single detector is in all likelihood rather high. This may be a practical consequence of Axelsson’s [12] result, suggesting that to retain a tractable false positive rate, the threshold for detection has been set rather high in these commercial anomaly detectors, thus increasing the false negative rate in tandem with the reduction in false positives, however in the absence of more details on the precise mode of operation and the numerical choices made in the algorithms, this remains speculative.

Issues with interpretability of anomaly IDS techniques are explicitly brought up in [11], in which they point out that many outlier detection methods “...[tend] not to provide semantic explanation as to why a particular instance has been flagged as an outlier.” The same observation within the context of IDS work has been made in some of the earliest work in anomaly-based IDS, including [33] (who in 1989 wrote “...explaining the meaning and likely cause of anomalous transactions ... must primarily be accomplished by a human.”) and [35] (extending and testing the earlier work of [34]) who note that, in the absence of automated analysis of audit data, a “security officer (SO) must wade through stacks of printed output.” They are also discussed briefly in [1], where the example of [37] is given, in which the titular question “Why 6?” (or, more completely, why a subsequence length of 6 turns out to be the minimum possible value that permits the anomaly IDS to find the intrusions in the test set) ultimately requires 26 pages and a full publication to answer in a satisfying manner. While this does not directly address operational issues, the point remains that if the researchers who study and implement a system require that much effort to understand why a single parameter must be set the way it is to generate acceptable performance, it is not a question that analysts responsible for minute-to-minute monitoring and analysis of network traffic are likely to have the time or inclination to answer.

At least part of the blame for the resilience of the semantic gap may be laid at the feet of the availability of data. Analysis methods must be developed to fit the available data, and regrettably the most commonly-used data set in intrusion detection research remains the infamous KDD’99 set, which began receiving criticism as little as 4 years after its initial release (see [38] and [39], for instance)

but—over the protests of many security researchers [24]—has been and continues to be widely used, particularly in the field of anomaly detection (see, e.g., [2, 9, 10, 22, 30, 32]).

The KDD’99 set is, superficially, extremely amenable to analysis by machine learning techniques (although see [38] and [39] for some notable issues); the data has 42 fields, the majority of which are continuously-valued, while the bulk of the remainder are Boolean. Only two fields, protocol type and service, are categorical, and both of these have a limited number of possible values (and, in practice, appear to be generally ignored in favor of continuous fields for most anomaly IDS research using this data). This enables and encourages an application of machine learning approaches that rely on metrics in inner product spaces (or their transformations) to anomaly IDS problems. The contextual information that human analysts often rely on to determine whether or not traffic is malicious (e.g., source and destination ports and IP blocks, protocol/port pairs, semantic irregularities such as malformed HTTP requests, and so forth) is very explicitly not included in this data, for the precise reason that it is very poorly suited to automated analysis by machine learning algorithms (recent work on language-theoretic security in fact suggests that automated analysis of many security problems is in fact *in principle* intractable, as it effectively reduces to solving the halting problem [40]).

The root of the semantic gap is thus in part the result of a vicious spiral: with the only widely used labeled data set being the KDD’99 data, which encourages the use of machine learning techniques that often rely on complex transformations of the data that tend to obscure any contextual information, the observation of [12] means that focus must be placed on reducing false positive rates to render them useful. This leads to more sophisticated and typically more complex methods, enlarging the semantic gap, leading to a greater emphasis on reduced false positive rate, and so forth.

4 The Base-Rate Fallacy, Anomaly Detection, and Cost of Misclassification

In addition to issues of interpretation, the simple fact that the overwhelming majority of traffic to most networks is not malicious has a significant impact on the reliability and usability of anomaly detectors. This phenomenon—the base-rate fallacy—and the impact it has on IDS systems is discussed in depth in [12]. We summarize selected relevant points of the argument here.

Briefly, the very low base rate for malicious behavior in a network leads to an unintuitive result, showing that the reliability of the detector (referred to in [12] as the “Bayesian detection rate”) $P(I|A)$, or the probability that an incident of concern actually has occurred given the fact that an alarm has been produced, is overwhelmingly dominated by the false positive rate of a detector $P(A| \sim I)$, read

as the probability that there is an alarm given that no incident of concern actually took place. A quick sketch by Bayes' theorem shows that:

$$P(I|A) = \frac{P(A|I)P(I)}{P(A|I)P(I) + P(A|\sim I)P(\sim I)}$$

where we assume all probabilities are with respect to some constant measure for a given site and IDS. If we then note that, by assumption, $P(I) \ll P(\sim I)$ for any given element under analysis, we can immediately see that the ratio on the right hand side is dominated by the $P(A|\sim I)P(\sim I)$ term. Since $P(\sim I)$ is assumed to be beyond our control, the only method of adjusting the reliability $P(I|A)$ is by adjusting the false positive rate of our detector.

This insight, has led to much work in recent results on reducing false positive rates in not just anomaly IDSs (see, e.g., [6] and [22]) but also in such misuse detectors as Snort [5, 18]. The availability of the KDD'99 set, which provides a convenient test-bed for such methods, despite its age and concerns about its reliability [38, 39], makes it relatively easy for research to focus on this issue, and so perhaps accounts for the popularity of KDD'99 despite its well-known issues.

The key insight from the base-rate fallacy argument is there is an extreme imbalance between the rate of occurrence of hostile traffic and that of malicious traffic, which in turn greatly exaggerates the impact of false positives on the reliability of the detector. However, if we extend the argument to include cost of classification, this same observation suggests a potential remediation. Assume that the cost of investigating any alarm, whether ultimately associated with an incident or not, is α , while the cost of not having an alarm when there is an incident (a false negative result) is β . We then have that

$$\begin{aligned} E[Cost] &= \alpha P(A) + \beta P(\sim A|I)P(I) \\ &= \alpha [P(A|I)P(I) + P(A|\sim I)P(\sim I)] + \beta P(\sim A|I)P(I) \end{aligned}$$

And if we once again assume that $P(I) \ll P(\sim I)$, then note that we can approximate then term $\alpha [P(A|I)P(I) + P(A|\sim I)P(\sim I)]$ by simply $\alpha P(A|\sim I)P(\sim I)$, such that:

$$E[Cost] \approx \alpha P(A|\sim I)P(\sim I) + \beta P(\sim A|I)P(I)$$

We assume that the cost of false negatives—classifying hostile traffic as benign—may be large enough that we cannot ignore the second term. Note that the expected cost in this case is now very sensitive to even small changes in the cost of examining alarms. This suggests that in the event that the false positive rate cannot be reduced significantly, we may still be able to operate the system at an acceptable cost by reducing the cost of examining IDS alerts. As we discuss later sections, current anomaly detectors are not well-suited to this task, and in fact attempting to reduce the cost by reducing $P(A|\sim I)$ have in fact inadvertently led to increases in α , by virtue of both the need to adapt the heterogeneous data available to IDS systems to standard machine learning techniques that focus on

inner product spaces, and the complex transforms of that data undertaken by these machine learning techniques in order to obtain accurate decision boundaries.

4.1 Cost Versus Threshold Tradeoffs

In many cases, the false positive rate and the false negative rate are related to each other; see, for instance, [6] where the effect of the support radius of a 1-class SVM on the true and false positive rate was examined. While shrinking the space mapping to a decision of ‘anomaly’ (or equivalently, increasing the detection threshold) clearly will reduce the false positive rate, it also decreases the sensitivity of the detector, and thus the true positive rate $P(A|I)$, and since $P(A|I) + P(\sim A|I) = 1$, i.e., every incoming packet must either trigger an alarm or not trigger an alarm, this must inevitably increase the false negative rate, potentially incurring a significant cost.

As a toy example, consider the Gaussian mixture problem, where our data comes from the following distribution:

$$f_X(x_i) = P(I)p(x_i; 1, 1) + (1 - P(I))p(x_i; 0, 1)$$

$$p(x_i; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x_i - \mu)^2\right\}$$

where for each x_i we must determine whether it was drawn from $p(x_i; 1, 1)$, in which case we deem it anomalous, or $p(x_i; 0, 1)$, in which case it is normal. Put another way, given some value x_i , our task is to decide whether $\mu_i = 1$ or $\mu_i = 0$. If we assume independence and construct a decision rule based solely on the current observation x_i , then we may fix a false negative rate of $P(\sim A|I) = p$; giving us a threshold for x_i of $x^\star = \Phi_{X|\mu=1}^{-1}(p)$ where Φ is the standard Gaussian CDF. From this we can obtain directly the false positive rate for this detector:

$$P(A|\sim I) = 1 - \Phi_{X|\mu_i=0}(x^\star)$$

Using this threshold-based decision rule, we have that the only possible way to decrease the false positive rate is to increase the threshold $x^\star$. As the CDF of x_i is by definition non-decreasing in x_i , we have immediately that increasing the threshold $x^\star$ will simultaneously decrease the false positive rate while increasing the false negative rate.

Turning to the cost, we have from above:

$$E[Cost] \approx \alpha P(A|\sim I)P(\sim I) + \beta P(\sim A|I)P(I)$$

$$= \alpha [1 - \Phi_{X|\mu_i=0}(x^\star)]P(\sim I) + \beta \Phi_{X|\mu_i=1}(x^\star)P(I)$$

And (in this example) can optimize with respect to cost by standard methods to find:

$$x_{(\text{opt})}^{\star} = [\ln \alpha P(\sim I) - \ln \beta P(I)] + \frac{1}{2}$$

Or more generally, letting f_0 denote the density function for normal traffic and f_1 denote the density of anomalous traffic, we have the usual form of the weighted likelihood ratio decision rule³:

$$\frac{f_1(x^{\star})}{f_0(x^{\star})} = \frac{\beta P(I)}{\alpha P(\sim I)}$$

At this threshold $x^{\star}$, any further increase in $x^{\star}$ leads to an increase in the expected cost of analysis, as the benefit of reduced false positives in the left hand term begins to be outweighed by the cost of increased false negatives.

Critically, for many IDS deployments, $P(I)$ may be completely unknown, and β is generally only roughly approximated. This immediately suggests that increasing the threshold for an anomaly detector may in fact be counterproductive, and in many cases it is difficult or impossible to know when precisely this tradeoff has occurred.

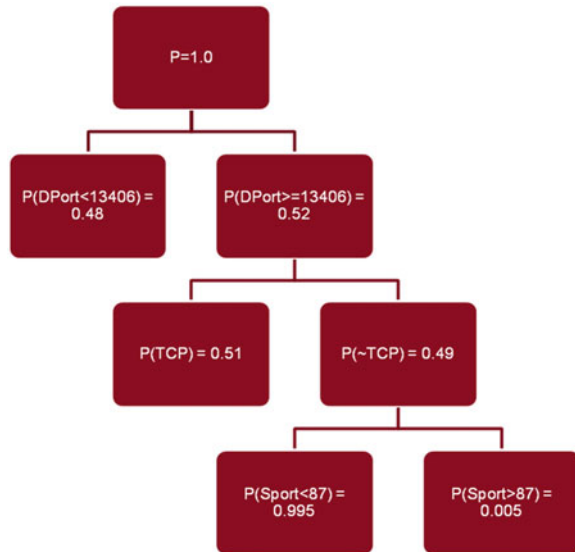
While this is—as in [12]—generally grim news for anomaly detection, the impact of α and β on the cost is also of interest. Security policies, segregating sensitive resources, and controlling physical access to the most critical systems and information may provide avenues to reduce β , however we defer this analysis to others. The dependence on α , however, is worth examining. Notice that—again—it is a multiplicative factor to the rate of normal traffic, suggesting that small adjustments in α can have an impact significantly greater than an equal adjustment in β . Indeed, $\frac{d}{d\alpha} E[Cost] \approx P(A|\sim I)P(\sim I)$ while $\frac{d}{d\beta} E[Cost] = P(\sim A|I)P(I)$ where again $P(I) \ll P(\sim I)$. This suggests that reducing the cost to analysts of examining and diagnosing alarms may provide significant benefit with respect to the cost of operation of an IDS. As we have seen above, efforts to control the false positive rate in much of the academic work relating to IDSs have led to increasingly complex classification systems. We contend that this trend has in fact significantly increased α , greatly reducing the utility of such systems for practical use.

³ Note another critical feature: if adversarial actors are capable of crafting their traffic to approximate f_0 , such that the quantity $\left|1 - \frac{f_1(x)}{f_0(x)}\right| \leq \epsilon$ for some small $\epsilon > 0$, and can control the rate of malicious traffic they send and hence $P(I)$, then they may craft their traffic such that the defenders have no $x^{\star}$ that satisfies the above relationship and so cannot perform cost-effective anomaly detection. We do not discuss this problem in detail, but reserve it for future work.

5 Revisiting Tree-Based Anomaly Detectors

As discussed previously, tree-based anomaly IDSs were among the first considered [33]. The work of [33] focused largely on a host-based approach to detecting misuse of computer systems on the basis of host actions and made efforts to use the tree structure to extract rules that could be both automatically applied to audit trail logs as well as directly interpreted by analysts. While the majority of recent anomaly IDS work has focused on various very popular and highly successful kernel methods that have been developed for machine learning in other areas, other more recent approaches have leveraged the work that has been done in random decision forests [28, 29] and begun to investigate their application to both supervised anomaly detection [10, 30] and unsupervised outlier detection [31, 32] in both batched and online modes. The fact that tree-based ensemble classifiers can achieve extremely high performance without further transforming the data into a more complex space to make it easier to separate (although note the empirical observation in [41] that the best performance is often obtained by including additional transformations of the covariates) and are typically quite robust to ancillary data [29, 31] make them attractive targets for anomaly detection algorithms, and initial results [10, 30–32]—albeit often on limited data—have shown that their performance is comparable or even superior to methods that employ more complex transformations.

While the semantic gap is not widely addressed in these papers, trees naturally lend themselves to extracting contextual information and classification rules that are generally much more interpretable to end users than the distance-based metrics of kernel methods. They also handle the heterogeneous data observed in networks more naturally than many other classifiers which either explicitly operate on features that map naturally to inner product spaces (see virtually any KDD’99-based paper) or require some transformation of the data to convert it into such a space (e.g., [23], transforming sequential bytes into n-grams). In particular, the most common form of splitting rule within continuous covariates in trees—axis-aligned learners—allow us to extract simple inequalities to define portions of rules relating to those attributes, and splits of ordinal variables allow us to extract relevant subsets from such fields. By tracing a path from the root to the final leaf node of any tree, it is extremely straightforward to extract the “rule” that led to the classification of the point in question in that leaf node; tabulation of the number of data points in the training data that fell into each internal node or leaf immediately gives us conditional probabilities based that may be immediately extracted from the data. Figure 4 shows a toy example of a decision tree based on the West Point/NSA CDX data [24] learned in an unsupervised manner similar to [32] (features selected at random from the available set, continuous features split at a randomly selected value from the ones available at that node, categorical features selected by information gain ratio; a maximum of three splits were permitted, and the splitting was terminated if the proposed split led to a leaf node of fewer than 100 points). A quick walk down the tree—taking the right-most split at each step—shows that we

Fig. 4 A toy decision tree

may construct the rule for an ‘anomaly’ learned by this tree as: “destination port $> 13,406$, not TCP, Source port > 87 ”. This rule is presented using the type of data that analysts work with on a daily basis (rather than elaborate RKHS representations, for instance), and the value of the rule may be assessed immediately.⁴

While single trees are straightforward to assess, combining rules in multiple trees is less straightforward. This question was touched on in [33] in the context of pruning the extensive database of generated rules, but only for strictly categorical data. They employ several criteria for pruning, beginning with a threshold function on the quality of the rule (favoring longer, more accurate rules with a smaller range of acceptable consequents), then removing rules where the predicates form a simple permutation of some other rule, next removing rules where the consequent matches some other rule and the predicates form a special case of some other rule, and finally pruning on number of exemplars of the rule in the data and depth. These criteria can be easily transferred to the case of continuous covariates.

⁴ In this case, any outgoing traffic to a relatively high destination port was deemed by an analyst to be unusual, but “certainly not a red flag”; the fact that it was non-TCP and did not originate from the lower end of the range of registered ports suggested a UDP streaming protocol, which often communicate across ephemeral ports; the analyst volunteered the suggestion that if it were in fact UDP it would likely not warrant further analysis. When the same analyst was presented with the outputs given in Fig. 1 through Fig. 3, they were of the opinion that it was not terribly useful, and that it not provide them with any guidance as to why it appeared suspicious; the semantic gap in action.

Table 1 Seven isolation tree rules and an associated meta-rule

Protocol	Transaction duration	Client port	Client packets	Client BPS	Service port	Service packets	Service BPS	$h(x)$
UDP	*	*	<69	*	*	*	*	2
*	*	*	<65	>0	*	*	*	2
UDP	*	*	*	*	*	*	>12,093	2
*	*	<23	*	*	*	*	>7,21,709	2
*	*	<137	*	*	*	*	>15,13,401	2
*	*	*	*	*	<3,268	>22	*	2
*	*	*	*	*	<4,985	*	>1,066	2
UDP	*	<23	<65	>0	<3,268	>22	>15,13,401	

Briefly, for all leaf nodes that lead to a classification of ‘anomalous’ in all trees (note that these nodes will vary depending on what kind of tree is used, either half-space trees as in [32], isolation trees as in [31], or some variety of density estimation tree as presented above and discussed in [29]), we extract the rules by walking the tree and identifying upper and lower bounds (if any) for the continuous fields, and the possible values (potentially including the special case of all values) for the categorical fields. We then attempt to identify the most compact set of aggregated rules for which the intersection of those sets is non-empty in all fields. As this process is equivalent to the set cover problem, we approximate the solution via a greedy algorithm, extracting the maximal set remaining in each iteration. Each aggregated rule is assigned a weight based on the number and weight of rules that are subsumed by it. In our example case, illustrated in Table 1, we used isolation trees of [31], simply taking their default values ($\psi = 256, t = 100, c(\psi) \approx 6.1123$). We can then define the weight of an aggregated rule as $s(x, \psi) = 2^{-\frac{\sum h_j(x)}{nc(\psi)}}$, as any packet that triggers our aggregated rule would, by definition, trigger all rules subsumed by the aggregated rule.

Note that the aggregated rule is extremely specific, represented in “natural units” of IDS data, and learned in an entirely unsupervised fashion from the data. In this case, analyst evaluation of the rule suggested that the primary feature of interest was the low port number in conjunction with the UDP protocol and the high rate of data from the external service. This suggested to the analyst a streaming protocol, consistent with many UDP transactions, but to a non-standard port. In addition, a feature that appeared to be irrelevant for this rule—the duration of the transaction—was naturally excluded, and could be easily trimmed from the display.

While this method takes an aggressive approach to rule pruning, essentially triggering only on instances that satisfy among the tightest constraints possible derived from the rule set represented by the trees, a similar approach could be used in a post hoc fashion, collecting only the subset of rules that the data point of interest triggered on its path through the forest, and performing a similar greedy aggregation method on them to determine aggregated rules that it satisfied. While this does not permit the one-time computation with subsequent amortization of the

time complexity that pre-processing of the forest permits, the fixed number of trees in the ensemble mean that the system may operate, similarly to the method of [32], with constant time complexity (albeit potentially a large constant).

While broader-scale testing on realistic data is necessary, and additional work to identify the most useful features is required, preliminary results indicate that the approach of using tree-based anomaly detection systems to identify and extract human-readable rules describing the generated anomalies is of great promise, and suggests that anomaly detection systems may well be able cross the semantic gap and see some degree of deployability in settings beyond the research lab.

6 Conclusion

We present an overview of many current approaches in the literature to anomaly-based intrusion detection, and examine in detail the issue of the semantic gap first articulated by [1]. This semantic gap is due in large part to the emphasis of anomaly detection research on outlier detection methods and machine learning techniques that have been designed primarily to operate within inner product spaces, which are not in general representative of the problem space for actual IDS deployment and operation. The result of [12] shows that the reliability of an anomaly detector—the probability that investigating an alarm will lead to detection of malicious or anomalous activity—is directly related to the false positive rate of the detector, and so much emphasis has been placed on reducing this false positive rate when treating the IDS as a black box. We extend this result with a cost analysis to show that in operational deployment, the interpretability of an IDS alert is also significant, and suggest that tree-based detection methods that operate in the untransformed space of IDS data not only are better-suited to the IDS problem space, but also lend themselves more naturally to human interpretation than high-dimensional kernel-based methods without the sacrifices in accuracy that have characterized categorical outlier detection methods. We finally show a proof-of-concept method for extracting rules from ensembles of trees, producing novel rules for detecting anomalous traffic built in a completely unsupervised fashion from live network data.

Appendix A

Random decision tree classification of KDD'99 data was performed using the Scikit-learn [25] package under Python 2.7.2 on a desktop commodity workstation. Training was performed using the file `kddcup.data_10_percent_corrected`, and testing was done on the file `kddcup.data.corrected`. 494,021 training records were used, and 4,898,431 test records. The three fields “Count”, “diff_srv_rate”, and “dst_bytes” were extracted along with the label field in both data sets; all other

A)	Guess_	Nmap	B)	Rookit	Warezcilent	C)	Pod	D)	Spy	ftp_	Phf	E)	Portsweep	Teardrop	Buffer_	Land	Imap	F)	Warezmater	Perl	Multihop	Back	Ipsweep	G)
Normal	passwd	Loadmodule				Smurf	Neptune		write						overflow								Satan	
A	0	53	2315	8	10	1020	971	264	5537	2	8	4	9558	752	28	20	12	5	3	7	2197	12480	950	
B	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
C	2210	0	1	0	0	0	0	9672	0	0	0	0	1	119	1	0	0	0	0	0	0	0	95	
D	402	0	0	0	0	0	40	0	0	0	0	0	42	108	1	1	0	0	0	0	0	0	1	
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	
F	3	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	
G	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Key: A) Normal B) Loadmodule C) Smurf D) Neptune E) Portsweep F) Warezmater G) Satan

data was discarded. The random decision forest was trained with the following parameters:

- Classification threshold: simple majority
- No bootstrapping used
- Features per node: 2
- Node splitting by informatin gain
- Minimum leaf samples: 1
- Minimum samples to split: 2
- Max tree depth: 9
- Number of trees: 11

Training the classifier required 4.4 s using a single processor, testing required approximately 122.8 s.⁵ The following confusion matrix was produced (note that we have omitted correct classifications on the diagonal for compactness, and that we have also omitted rows corresponding to predictions that the classifier never produced).

Total false negatives: $36204/4898431 \approx 0.007$

Total false positives: $2622/4898431 \approx 0.0005$

The most common errors were misclassification of the IPsweep attack as normal traffic, and classification of flows corresponding to the Neptune attack as the Smurf attack. Random inspection of the IPsweep misclassifications suggests that each “attack” had several records associated with it; while many individual records were not correctly labeled, all instances that were examined by hand had at least one record in the total attack correctly classified. As the Smurf and Neptune attacks are both denial of service attacks, some confusion between the two is to be expected.

While these results certainly demonstrate that random decision forests are accurate and efficient classifiers, the alternative that the KDD’99 data is simply not a terribly representative data set for IDS research should not be excluded.

References

1. R. Sommer, V. Paxson, Outside the closed world: on using machine learning for network intrusion detection,” in *2010 IEEE Symposium on Security and Privacy (SP)*, 2010
2. P. Laskov, P. DÄessel, C. SchÄfer, K. Rieck, in *Learning Intrusion Detection: Supervised or Unsupervised?*, ed. by F. Roli, S. Vitulano (Springer, Berlin, 2005), pp. 50–57
3. M. Roesch, Snort – lightweight intrusion detection for networks, in *Proceedings of the 13th USENIX Conference on System Administration*, 1999, pp. 229–238

⁵ Due to the large size of the test set, it was not loaded into memory all at once, and instead was read sequentially from disk. Total time elapsed was 1023.3 s, of which profiling indicated that roughly 88 % was consumed by disk I/O operations. As our interest was in offhand comparison and not production use, we did not attempt to optimize this further.

4. V. Paxson, Bro: a system for detecting network intruders in real time. *Comput. Netw.* **31**(23–24), 2435–2463 (1999)
5. J. Long, D. Schwartz, S. Stoecklin, Distinguishing false from true alerts in Snort by data mining patterns of alerts, in *Proceedings of 2006 SPIE Defense and Security Symposium*, 2006
6. M. Sato, H. Yamaki, H. Takakura, Unknown attacks detection using feature extraction from anomaly-based IDS alerts, in *2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet (SAINT)*, 2012
7. Y. Song, M.E. Locasto, A. Stavrou, A.D. Keromytis, S.J. Stolfo, On the infeasibility of modeling polymorphic shellcode – Re-thinking..., in *MACH LEARN*, 2009
8. H. Debar, M. Dacier, A. Wespi, Towards a taxonomy of intrusion-detection systems. *Comput. Netw.* **31**(8), 805–822 (1999)
9. O. Depren, M. Topallar, E. Anarim, M.K. Ciliz, An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert Syst. Appl.* **29**(4), 713–722 (2005)
10. J. Zhang, M. Zulkernine, A. Haque, Random-forests-based network intrusion detection systems. *IEEE Trans. Syst. Man Cybern. C Appl. Rev.* **38**(5), 649–659 (2008)
11. N. Abe, B. Zadrozny, J. Langford, Outlier detection by active learning, in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, 2006
12. S. Axelsson, The base-rate fallacy and the difficulty of intrusion detection. *ACM Trans. Inf. Syst. Secur.* **3**(3), 186–205 (2000)
13. A. Koufakou, E.G. Ortiz, M. Georgiopoulos, G.C. Anagnostopoulos, K.M. Reynolds, A scalable and efficient outlier detection strategy for categorical data, in *19th IEEE International Conference on Tools with Artificial Intelligence, 2007. ICTAI 2007*
14. M.E. Otey, A. Ghoting, S. Parthasarathy, Fast distributed outlier detection in mixed-attribute data sets. *Data Min. Knowl. Discov.* **12**(2–3), 203–228 (2006)
15. X. Song, M. Wu, C. Jermaine, S. Ranka, Conditional anomaly detection. *IEEE Trans. Knowl. Data Eng.* **19**(5), 631–645 (2007)
16. C. Cortes, V. Vapnik, Support-vector networks. *Mach. Learn.* **20**(3), 273–297 (1995)
17. K. Wang, S. Stolfo, One-class training for Masquerade detection, in *Workshop on Data Mining for Computer Security*, 2003
18. R. Perdisci, G. Gu, W. Lee, Using an ensemble of one-class SVM classifiers to harden payload-based anomaly detection systems, in *Sixth International Conference on Data Mining, 2006. ICDM'06*. 2006
19. S. Mukkamala, G. Janoski, A. Sung, Intrusion detection using neural networks and support vector machines, in *Proceedings of the 2002 International Joint Conference on Neural Networks*, 2002
20. J. Weston, C. Watkins, Technical Report CSD-TR-98-04, Department of Computer Science, *Multi-class Support Vector Machines*, Royal Holloway, University of London, 1998
21. R. Chen, K. Cheng, Y. Chen, C. Hsieh, Using rough set and support vector machine for network intrusion detection system, in *First Asian Conference on Intelligent Information and Database Systems*, 2009
22. T. Shon, Y. Kim, C. Lee, J. Moon, A machine learning framework for network anomaly detection using SVM and GA, in *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, 2005
23. K. Wang, S. Stolfo, Anomalous payload-based network intrusion detection, in *Recent Advances in Intrusion Detection*, 2004
24. B. Sangster, T. O'Connor, T. Cook, R. Fanelli, E. Dean, J. Adams, C. Morrell, G. Conti, Toward instrumenting network warfare competitions to generate labeled datasets, in *USENIX Security's Workshop on Cyber Security Experimentation and Test (CSET)*, 2009
25. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher,

- M. Perrot, E. Duchesnay, Scikit-learn: machine learning in python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
26. P. Biondi, Scapy, a powerful interactive packet manipulation program. , *Scapy*, 2011, <http://www.secdev.org/projects/scapy/>
27. V. Frias-Martinez, J. Sherrick, S.J. Stolfo, A.D. Keromytis, A network access control mechanism based on behavior profiles, in *Computer Security Applications Conference, 2009. ACSAC'09. Annual*, 2009
28. L. Breiman, Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
29. A. Criminisi, J. Shotton, E. Konukoglu, *Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning*, Microsoft Technical Report, 2011
30. D.S. Kim, S.M. Lee, J.S. Park, *Building Lightweight Intrusion Detection System Based on Random Forest*, ed. by J. Wang, Z. Yi, J.M. Zurada, B. Lu, H. Yin (Springer, Berlin, 2006), pp. 224–230
31. F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation-based anomaly detection. *ACM Trans. Knowl. Discov. Data* **6**(1), 3:1–3:39 (2012)
32. S.C. Tan, K.M. Ting, T.F. Liu, Fast anomaly detection for streaming data, in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, vol. 2, 2011
33. H.S. Vaccaro, G.E. Liepins, Detection of anomalous computer session activity, in *Proceedings of 1989 IEEE Symposium on Security and Privacy*, 1989
34. D.E. Denning, An intrusion-detection model. *IEEE Trans. Softw. Eng.* **13**(2), 222–232 (1987)
35. M. Mahoney, P. Chan, An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection, in *Recent Advances in Intrusion Detection*, 2003
36. J. McHugh, Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Trans. Inf. Syst. Secur.* **3**(4), 262–294 (2000)
37. T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. Jalali, P. Neumann, H. Javitz, A. Valdes, T. Garvey, *A real-time intrusion-detection expert system (IDES)*, SRI International, Computer Science Laboratory, 1992
38. M. Molina, I. Paredes-Oliva, W. Routly, P. Barlet-Ros, Operational experiences with anomaly detection in backbone networks. *Comput. Secur.* **31**(3), 273–285 (2012)
39. K.M. Tan, R.A. Maxion, “Why 6?” Defining the operational limits of stide, an anomaly-based intrusion detector, in *Proceedings of the IEEE Symposium on Security and Privacy*, 2001
40. L. Sassaman, M.L. Patterson, S. Bratus, A. Shubina, The Halting problems of network stack insecurity, in *USENIX*, 2011
41. Z. Zhou, *Ensemble Methods: Foundations and Algorithms* (Chapman & Hall, 2012)



<http://www.springer.com/978-1-4614-7596-5>

Network Science and Cybersecurity

Pino, R.E. (Ed.)

2014, X, 285 p., Hardcover

ISBN: 978-1-4614-7596-5