Detecting Novel Scans Through Pattern Anomaly Detection

Alfonso Valdes, SRI International
Valdes@sdl.sri.com

Abstract

We introduce a technique for detecting anomalous patterns in a categorical feature (one that takes values from a finite alphabet). It differs from most anomaly detection methods used to date in that it does not require attack-free training data, and it improves upon previous methods known to us in that it is aware when it is adequately trained to generate meaningful alerts, and it models data not as normal and anomalous but as falling into one of a number of modes discovered by competitive learning. We apply the technique to port patterns in TCP sessions (the alphabet being the port numbers) and highlight interesting patterns detected in simulated and real traffic.

We propose extensions where the learned pattern library can be seeded and some patterns of interest can be labeled, so that certain patterns generate an alert no matter how frequently they are observed, while others labeled benign do not generate alerts even if rarely seen. Finally, we outline a hybrid system approach to closely integrate anomaly and misuse detection, arguing that the historical dichotomy with which many researchers approach these techniques is now artificial.

Acknowledgments

The work presented in this paper was funded by the Defense Advanced Research Projects Agency (DARPA), under subcontract to BBNT Solutions LLC, government contract number F30602-98-C-0012. EMERALD is a trademark of SRI International.

Introduction

Papers in intrusion detection frequently draw a distinction between anomaly and misuse detection. The usual discussion cites the advantage of one or the other (generalization to novel attacks for the former, specificity and lower false alarms for the latter) and then presents an application to a simulated or actual dataset. To date, systems that have enjoyed some success tend to be of the misuse detection class, and almost all are signature based. Anomaly detection systems have an inherent limitation in that they detect the anomalous, which may not be malicious. They thus have a built-in false alarm mechanism, namely, at best they will generate alerts equal to that fraction of some empirical distribution function that a security administrator considers worthy of interest. Some anomaly detection systems attempt to characterize all normal system behavior and thereby reduce this fraction to zero. Learning normal behavior imposes the need for training on anomaly detection systems.

Cognizant of the limitations and historical record of anomaly detection, we nonetheless feel such systems have their place, although seldom if ever as the only detection mechanism guarding a traffic stream. We have developed a competitive learning technique to detect anomalous patterns in network data, and closely coupled this anomaly detector with a probabilistic misuse detection engine. Although at present we examine anomalous patterns of port usage in TCP sessions, the technique is more generally applicable. Our method addresses three historical problems with anomaly detection:

- Rather than assuming that traffic is bimodal (normal or anomalous), our approach models data as belonging to one of several modes. The number of modes is not known beforehand. Detections are based on the rarity of the mode that matches an observed sequence. We also propose an approach where we may alert on malicious modes even if they are not particularly rare.
- Previous anomaly detection systems have required training on data known to be attack free or in which attack sequences are labeled. For live traffic, this is very difficult to obtain. Our approach will detect attacks that fall into particular modes that may be rare or explicitly labeled as malicious. Seeing attacks in the training data merely reinforces the modes to which the attack belongs.
- Given a detection threshold, the system is aware of when it has seen enough sequences to consider itself trained. There is no fixed training interval as such. With TCP traffic, we obtain useful detection within hours of a clean start.

The remainder of this paper is organized as follows. In the next section, we discuss related work in anomaly-based intrusion detection, as well as work in the fields of clustering, neural networks/competitive learning, and data mining. Although we are addressing anomaly-based intrusion detection, the lineage of our approach draws more from the other fields. We then describe the particular instantiation of our method to detection of anomalous port patterns in TCP sessions. We provide results on simulated and actual network data. In our discussion of future work, we present the concept of seeding and labeling the library of observed modes, so that our detector will always alert on known or learned malicious behavior modes. We also introduce a Bayes framework to integrate probabilistic misuse detection, signature systems, and anomaly detection to exploit the strengths of each. It is thus argued that the dichotomy between anomaly and misuse detection is unnecessarily limiting.

Related Work

IDES and NIDES [Jav91] introduced anomaly detection to the field of intrusion detection, using a statistical profile learning approach. The features considered by NIDES were related to user activity as observable in audit data, and included continuous (such as CPU usage) and categorical features (such as file usage). Usage rate or intensity features derived from interevent times were also used. Deviations of the short-term profile about the learned long-term profile were summarized in a chi-squared-like statistic, for which the system learned an empirical distribution rather than referring to standard tables. To control state space explosion, the system included exponential fading memory to forget extremely rare observations.

The influential work of Forest [For96] used ordered sequences, or n-grams, of commands or system calls to distinguish user and program activity. Warrender [War99] examined more sophisticated models applied to the same sequence data, such as hidden Markov models (HMM), but did not show dramatic improvement over the original work of Forest. Maxion [Max02] has extensively studied the theoretical efficacy of these approaches, considering such factors as the underlying entropy in the data sequences. Researchers at IBM-Zurich built on the sequence analysis technique by adapting a proprietary method for gene sequence analysis [Wes00]. This method is not limited to fixed-length sequences and can accommodate sequences where a limited number of intervening alphabet tokens are observed.

The SPADE and SPICE work [SD02] attempts to detect novel portscans by observing anomalous port pattern activity, as we do here. Their approach is to maintain empirical probabilities of source/target pairs, whereas we are concerned principally with target ports, which greatly reduces the size of state the system must maintain. Moreover, as we shall see our technique considers port patterns as well as single ports, achieving a degree of correlation in the SPICE sense.

One historical limitation of anomaly detection to date is the assumption that the domain can be classified into two modes. The very name "anomaly detection" implies some means of specifying or learning "normal" and its complement as "anomalous" and therefore worthy of alert. Forest's terminology classifies sequences into "self" and "non-self," based on the immune system metaphor, retaining the notion of a bimodal dichotomy. On some level, the library of n-grams can be considered a polymodal knowledge base, and the normal/anomalous dichotomy is imposed over this, namely, that which is not in the library is anomalous.

Related to this limitation is the need for training data known to be attack-free or with the attacks labeled. Eskin et al. [Es02] demonstrate effective geometric-distance-based techniques for anomaly detection that do not require labeled data. This work shares with the present development the notion of multimodality or natural clusters in the data. Their methods differ from our work in that it is still assumed that intrusions are rare events and thus detectable as outliers in a feature space, although their result appears to permit alert triggering based on cluster membership rather than cluster rarity, as we propose below.

Recent work by Mahoney [Mah02] shares with the present development the concept of triggering not just on novelty but on rarity as well. Mahoney [Mah02] does not ostensibly use labeled data, but the approach relies heavily on source IP as a discriminating feature, which is a very effective discriminator in the data set they considered. In Table 6 of [Ma02], for example, it appears that source IP is responsible for at least as many detections as all other features combined. This data set is based on the Lincoln data [Lip00]. While it is true that some source IP addresses only interact with destinations through various forms of attack, this effect is over represented in the Lincoln data. In many sets of real world data we have examined, source IP is a far less reliable attack label.

Autoclass [Ch96] also uses an approach that attempts to learn natural clusters in data when the number of clusters and cluster membership are unknown. It employs an iterative batch training procedure similar to the Expectation/Maximization (EM) technique of Dempster [De77].

Our technique does not require a prior batch training step since it trains on line and is self aware of the point at which alerts can be triggered a configurable anomaly threshold.

Our objective is to characterize clusters of anomalous activity and while we alert primarily on the rarity of the cluster, we are able to label a cluster as potentially intrusive (even if it occurs more frequently than the anomaly threshold) or conversely label a rare cluster as not intrusive.

There have been two approaches to training anomaly detection. In NIDES and n-gram analysis, the system learned normal behavior by observing a large and varied set of data known or believed to be free of attacks. For real data, guaranteeing completeness and the absence of attacks is extremely difficult. The IBM Zurich group trains its detector by collecting sequences obtained when running the test suite of the program to be monitored. While not proving that this is theoretically complete, this is a reasonable approach and likely to exercise rare usage modes that may not be observed by live traffic analysis over a short time interval.

The present work departs from these approaches in assuming that usage falls into a number of modes, which can be represented as patterns in a pattern library. The number of modes is not known beforehand, nor is labeled training data required. The pattern library approach is reminiscent of n-grams, although we consider the presence or absence of symbols, or their relative frequency, and not their sequence. Alerts are generated not based on whether a pattern is or is not in the library, but on how rare it is. As such, there is no fixed training interval required; the system recognizes when it has built an adequate probability distribution over the pattern library. Moreover, the training data need not be labeled or attack free. Ideally, attacks will form distinct nodes, and will generate alerts if they are sufficiently rare. In future work, we will seed the library with labeled patterns, enabling alerts for malicious patterns even if they are not particularly rare.

In supervised clustering, a set of observations is labeled as belonging to a particular class, and a classification algorithm selects some function of the features that achieves separation of the classes. This function can be a decision function in the sense of discriminant analysis, or the connective weights of a neural classifier. The cluster label can be the putative user, or it may represent a mode of usage (such as all the activity attributable to a particular application window).

We also consider unsupervised cluster approaches such as k-means and adaptive resonance theory (ART [Gro88]). The first method “seeds” a set of cluster centroids and then iteratively assigns observations to the centroids, subject to some limits on the number of centroids allowed and optimizing a goodness of fit criteria for centroid membership.

Discriminant functions and k-means are classical techniques that make strong assumptions about the separability of the clusters in high-dimensional space, whereas neural networks (to include ART in this context) make fewer such assumptions and empirically achieve more arbitrary separation.

The clusters identified by unsupervised methods may indicate natural usage modes that may or may not be associated with particular users.

Pattern Recognition and Competitive Learning

Pattern recognition and competitive learning techniques [Tou74, Rum88] commonly represent an observation as a binary feature vector, where an entry is 0 or 1 depending on whether or not the corresponding alphabet token was present in the observation. A candidate observation is then evaluated against a library of existing observations, and is classified as belonging to the library exemplar that best matches it. This matching is done by similarity functions such as the dot product, that is,

$$Sim\left(W^j, Y\right) = W^j \bullet Y$$
$$= \sum_{i=1}^{n} w_i^j y_i$$
$W^j =$ Weight vector for library class exemplar j.

This similarity function has the desirable intuitive property that, for binary feature vectors, it is a count of the number of features in which X and Y agree. It is also amenable to implementations that are computationally efficient. Often, the similarity is divided by the total number of alphabet tokens in either pattern, so that it is a floating point number between 0 and 1 inclusive, with 1.0 corresponding to a perfect match.

In the typical pattern recognition application, there are a number of exemplars for which the classification is known. Ideally, these form clusters of patterns with very high within-cluster similarity and low between-cluster similarity. A training algorithm identifies important features in a cluster, generating a library of class prototype patterns. New patterns are then presented, and the system classifies these into one of the library classes according to what it has learned in the training phase. The empirical validation of the approach for an application of interest is how well the method classifies new patterns not used by the system in training, perhaps moderated by non-uniform misclassification costs.

Competitive learning further extends this in two respects. A class prototype is continuously maintained as a weight vector, and the existing classes "compete" for each new pattern. In this case, a slightly different similarity function, still based on the dot product, is employed. In ART terminology, we classify a pattern as belonging to a particular class if it "resonates" adequately with the stored exemplar of that class, which then adaptively learns from the new observation.

$$Sim\left(W^j, Y\right) = W^j \bullet Y$$
$$= \sum_{i=1}^{n} w_i^j y_i$$
$W^j =$ Weight vector for library class exemplar j.

Note that while this is still based on the dot product, it is now the dot product of a floating-point weight vector with a binary vector. The class that "wins" then adapts its prototype slightly in the direction of the new pattern, where "slightly" is typically based on an annealing schedule that allows greater adaptation early in the training. The actual nature of the adaptation is to increase slightly the weights corresponding to entries where the new pattern has a value 1 (denoted the "active input lines" in [Rum88]). All weights are positive and sum to unity.

Finally, we may consider systems that dynamically grow new library classes if none of the ones currently defined are sufficiently similar to the new pattern. Such systems can start with an empty library, and work for data sets for which the label is not known. Our system is of this type, and seeks to discover the number of clusters into which the data appears to be organized. Again, there is an analogy to ART. In ART, if the observed pattern does not adequately resonate with any stored exemplars (based on a stress function), it becomes a new exemplar. This is a similar approach if the underlying mathematics are somewhat different

Data Mining

Data mining approaches [Agr88] seek to discover association rules that describe the degree to which certain types of items tend to occur in the same transaction. A common example used in the data mining literature is that of a customer purchase, or transaction, where the items purchased in the particular transaction are the features of interest. Although the language and discovery algorithms are somewhat different, there is a commonality in the representation of data with the binary patterns described above. We can consider a transaction to be a binary pattern vector in the sense previously described, and the items as binary features (the entry in the feature vector corresponding to a particular item is 1 if the item is purchased in the transaction). The alphabet is the universe of items available for inclusion in the basket, and the basket itself corresponds to a pattern. Data mining seeks association rules that estimate the expectation of observing a particular item in a transaction given the occurrence of a particular itemset in the transaction. The technique is also interested in discovering large itemsets, typically by an off-line learning procedure involving multiple passes through the entire database of transactions

These techniques can be applied to other domains, such as pattern classification. In this case, we can use association rules where *X* is a library pattern and the itemset in the consequent of an association rule is a binary indicator of the class to which the pattern belongs. For example, Lee [Lee99] uses such an approach to build computer intrusion detection models, where the association rules match patterns to intrusions of certain types.

The concept of itemset discovery is analogous to competitive learning. We consider two market baskets, with items selected from an "item alphabet". A common formula for the similarity of two baskets counts the items in common (the intersecting item set) and divides by the total number of items in both baskets (the union).

$$Sim(X,Y) = \frac{N(X \cap Y)}{N(X \cup Y)}$$

X,Y are the itemsets

$N(.) =$ element count

This is merely a restatement of the dot-product similarity function presented earlier for binary patterns, here expressed in set notation.

Port Pattern Anomaly Detection

Our model assumes that there is a categorical feature (a feature taking values from some alphabet) observed an arbitrary number of times in some suitably aggregated analysis unit of data such as a session. In the context of TCP sessions, we have chosen the ports invoked as the feature of interest. The "market basket" or "itemset" is a TCP session, defined as a temporally contiguous burst of traffic from the same source, as defined in [Va00], which introduced a TCP Bayes network intrusion detection sensor. We have enhanced this sensor to include a port pattern anomaly detection. Although port pattern anomalies are the initial instantiation of this technique, it is more general in its applicability. In particular, we believe the technique is applicable to TCP flag combinations and system call anomalies, and hope to explore these applications in the future.

A pattern is a set of symbols taken from the alphabet. It may be of arbitrary length. We represent the pattern as the set of symbols between brackets […], enumerating the symbols observed or including a hit count as well. Under the first approach, the pattern might be [C, F, M], whereas under the second the representation might be [C:100, F:1:, M: 99]. We do not at present consider the sequential order of the symbols, as is done in the work of Forest and Wespi. Library patterns are stored as probabilities, and newly observed patterns are normalized to probabilities as well. This permits the update procedure below to treat the new pattern as an observation with an effective count (for our purposes, the mixing weight) of 1.0.

When patterns are stored as probabilities, we adapt the similarity function expressed above in terms of intersections and unions as follows. The intersection (numerator) term is the sum of the minimum probabilities for those alphabet tokens where the patterns overlap. The union (denominator) term is the sum of the maximum probability for those alphabet tokens seen in either pattern (note that this can exceed unity). This is illustrated in the following numerical example.

$$X = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} \end{bmatrix}$$

$$Y = \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 \end{bmatrix}$$

Patterns overlap in the first two entries.

Y is minimum probability.

$\Rightarrow$ Numerator $= {}^{2}\!/_{5}$

X is maximal probability in the first, second, and sixth entries.

Y is maximal elsewhere.

$\Rightarrow$ Denominator $= {}^{3}\!/_{3} + {}^{3}\!/_{5} = {}^{8}\!/_{5}$

$$Sim(X,Y) = \frac{{}^{2}\!/_{5}}{{}^{8}\!/_{5}} = \frac{1}{4}$$

The system maintains a library of patterns that may be initially empty, or it may be seeded as described under future work below. When we observe a pattern, we evaluate its similarity with respect to patterns in the library. If it matches one or more stored patterns above a configurable threshold, then the new pattern is considered to belong to the class of the best matching exemplar. Borrowing terminology from competitive learning, we call the pattern achieving the best match the winning pattern or simply the winner.

Algorithm to pick winner :

Find K s.t.

$$Sim(X, E_K) \geq Sim(X, E_k) \forall k$$

X = observed pattern

E_k = kth pattern exemplar in library

If $Sim(X, E_K) \geq T_{match}, E_K$ is the winner

Else insert X into the library of pattern exemplars

T_{match} = Minimum match threshold

The winner is adaptively modified by mixing in the new pattern. The degree of mixing depends on the historical count of observations of the library exemplar. This count is exponentially decayed with a slow aging factor. Patterns that are frequently seen are therefore less perturbed by a new observation.

$$E_K \leftarrow \frac{1}{n_K + 1}(n_K E_K + X)$$

$n_K =$ Historical (possibly aged) count of observances of E_K

Whether the observation triggers an alert depends on the normalized probability of the winner. In fact, as in the NIDES system, the anomaly score is the tail probability, that is, the sum of probabilities of all patterns that are as probable or less probable than the winner. If this anomaly score is sufficiently close to zero, an alert is generated.

$\Pr(E_K)$ = Historical probability of pattern K

$$= \frac{n_K}{\sum_k n_k}$$

$Tail\_\Pr(E_K)$ = Historical tail probability of pattern K

$$= \sum_{\Pr(E_k) \geq \Pr(E_j)} \Pr(E_j)$$

If $Tail\_\Pr(E_K) \leq T_{alert}$, generate alert

T_{alert} = alert threshold

The need to consider tail probability rather than the raw pattern probability is motivated by the following example. Suppose we have 1,000 patterns in the library, all of which are equally probable with historical probability 0.001. However, they all have a tail probability of 1.0. In this pathological case of a maximum entropy pattern library, there is no meaningful alert threshold. Typically, we observe that a few of the patterns account for most of the probability. In our environment, for example, web and mail account for 95% of the port patterns.

We track the empirical probability of adding new patterns by the above algorithm. If it is fairly common to see new patterns, the technique is of limited utility.

An immediate concern is that the length of the alphabet can lead to a state space explosion. For example, considering TCP ports, the alphabet contains over 1000 possible values. Since we do not limit pattern length, it is apparent that the number of unique patterns can be arbitrarily large. The pattern anomaly approach can work only if the actual number of observed patterns is comparatively small.

The mechanisms for intelligent forgetting are similar to the category dropping heuristics introduced in our earlier NIDES work. We invoke two forms of forgetting. First, if after an update operation the normalized hit count (or equivalently, category probability) of an entry in the pattern drops below some threshold, that entry is dropped from the pattern. Dropping these entries does not significantly affect that pattern's ability to resonate with a new pattern, since the entry dropped is by definition extremely rare. For this mode of forgetting, we require that patterns be represented with associated hit counts. We also forget entire patterns whose normalized probability is extremely rare. Using these techniques, over weeks of live traffic analysis, the pattern library grows to about 80 unique patterns.

We can thus sum up the conditions under which the technique can be reasonably successful.

- The learned pattern library is modal rather than high entropy (for the number of patterns learned).
- After an initial transient, learning of new patterns is infrequent.
- The total number of patterns is small relative to the possible number of patterns given the cardinality of the alphabet.

We observe that these conditions hold satisfactorily with TCP port patterns, if we eliminate patterns not likely to be interesting and employ intelligent forgetting. We have taken steps to avoid evaluating patterns not likely to be interesting, such as those observed from FTP mget sessions or those otherwise dynamically assigned by the system.

Because of the nature of the pattern similarity function, there is generalization potential to detect variants of malicious patterns.

Results

We have implemented the above procedure examining the list of ports hit in TCP sessions (an observation is the list of ports "hit" in a session, and the "hit count" for each), as well as for a synthetic measure with six categories describing the TCP session open and close result. This runs concurrently with our eBayes-TCP session monitor that we have previously described [Va00]. It is a misuse detector with a knowledge base encoded as hypotheses and conditional probability relations rather than signatures. The integrated system alerts if either the TCP session has sufficient posterior probability of membership in a misuse class or if the port pattern is anomalous. If both conditions are true, the misuse alert overrides the anomaly alert.

We have examined the Lincoln Laboratories 1999 Intrusion Detection Evaluation Data [Lip00]. In the fourth and most difficult week of that data, Bayes-TCP finds 13 attacks, including several stealth attacks. In addition, the anomaly procedure identifies 11 unusual port patterns at the 0.5% threshold. All correspond to attacks. Again, we remind the reader that the anomaly detection does not consider "labeled" data and does all its learning on the fly.

We run this system against our laboratory’s internet gateway. For real-world data, ground truth is not known, but it is the opinion of our system administration staff that many of the port anomaly alerts are genuine attacks, and many more are at least worthy of attention. We also see alerts for activity that is rare but probably benign in our environment, such as https and ssh. The following table lists some discovered anomalous port patterns. Many are singletons (a single rare port) but there are some interesting combinations of ports for which we have no prior signature. In the coupled Bayes/anomaly system, the Bayes alert overrides the anomaly alert, so that some of the patterns that would generate an anomaly alert based on pattern rarity cause a Bayes alert instead. For these patterns, the comment field contains the phrase “portsweep override”. We have excluded obvious vertical port sweeps that fall into this category as well.

Discovered Rare Pattern	**Comment**
[119]	Happy99 Trojan
[139], [135, 139]	Netbios
[22], [443], [53], [80, 443]	Probably benign but rare
[27374]	Sub7
[51506]	?
[524]	NCP
[6346]	gnutella
[636]	ldaps
[80, 39, 445]	Port sweep override, 4 invalid IP, web, nebios, Win2k server message block
[9, 12345, 27374, 139]	Portsweep override, Netbus, Sub7, Netbios

The Lincoln data is characterized by about 30 port patterns. The live data falls into 80 port patterns over a period exceeding a month.

Independently, a high-traffic set of real world data resulted in 63 patterns over a period slightly over a day, representing more than 130,000 TCP sessions. Of these, the patterns [80] and [25] combined for about 92% of the total sessions. This validates our assumption that, at least in this domain, what is observed only sparsely populates what is possible, and therefore this procedure is applicable.

Future Work

Alert Triggering and Patern Seeding

As presently implemented, our port pattern anomaly component is a pure anomaly detector, which means it generates alerts on activity it considers extremely rare. It has been pointed out [McH01] that anomalous activity is not necessarily intrusive, and intrusions are not necessarily anomalous. In particular, sufficiently rare but non-malicious activity will trigger alerts. As we mentioned above, https and ssh fall into the anomalous but probably benign category in our site.

Moreover, patterns that are likely to be malicious but are regularly observed (e.g., as a result of a widely available scripted attack) may not be sufficiently rare to trigger alarms.

Other patterns may require a policy-based response. For example, we observe that our system discovered gnutella, initially as an anomalous port. We may conjecture that, with increased use of this service, the port may be observed with enough frequency that it does not trigger an anomaly alert. The administrator may choose to include this port in the seeded pattern library with an alert/no alert label depending on policy.

Pattern library exemplars are instantiated as objects which contain, among other things, a trigger tag. This tag assumes the values ALERT_IF_RARE, ALERT_ALWAYS, and ALERT_NEVER. Pure anomaly detection is equivalent to the tag assuming the value ALERT_IF_RARE, and this is the default for newly allocated cluster objects.

We will enhance our system in the near term to incorporate a seeded pattern library. In such a library, the patterns corresponding to https and ssh would have the tag set to ALERT_NEVER. This prevents detection of attacks on these ports as anomalies, but does not preclude detections from a protocol-specific IDS. Conversely, ports corresponding to gnutella may be represented as objects in which the tag is set to ALERT_ALWAYS.

The security administrator would seed new patterns and change the alert/ trigger dynamically in response to new observations or alert advisories.

Hybrid Systems

Papers in intrusion detection frequently draw a dichotomy between signature systems and anomaly detection. The Bayes TCP system of [Va00] introduces a probabilistic model-based system, where models of normal and malicious use are represented as conditional probability relations rather than very specific signatures. The port pattern anomaly system presented here is presently integrated as a component of Bayes-TCP.

A Bayes framework can easily incorporate a Bayes subsystem (a Bayes subtree is itself a Bayes tree). Pattern anomalies can be integrated either as hypotheses in themselves (the observed pattern is unusual, or malicious if it correlates with a seeded and labeled pattern). Alternately, an anomaly subsystem can condition prior probabilities elsewhere in the Bayes system (for example, the rarity of the pattern makes the system more suspicious by increasing the prior probability of certain malicious usage modes).

A Bayes-like framework can in fact be used to integrate rule based systems as well. Rules can be implemented as nodes that establish or refute hypotheses about a sequence of traffic. This can be done with certainty or with perturbations to express uncertainty in the underlying observation. As a practical observation, such perturbations prevent computational pathologies that arise as the result of hard contradictions, although if these arise obviously the system's rule base should be examined. The appendix outlines a mathematical approach to implement rule-based subsystems into an overall Bayes framework.

A Bayes-like framework can thus provide a hybrid system approach to tightly integrate Bayes, rule-based, and anomaly systems. As such, we believe the dichotomy between rule-based and anomaly systems is artificial

Summary

We have presented a technique to discover anomalous patterns in certain classes of categorical data. Rather than formulating the problem in a bimodal framework (learn normal and identify anomalous as that which is not normal) our approach atempts to discover underlying clusters of patterns. It is hoped that these clusters of patterns correspond to behavioral modes in the data.

Our technique adapts competitive learning and data mining approaches to discover these modes dynamically. This eliminates the need for attack-free or labeled training data.

The system is aware of the point at which it can declare a pattern anomalous at some configurable threshold. It does not require the user to guess at a sufficiently long (in time) training interval. In realistic traffic, the system trains to a useful degree in a matter of hours at most.

At present, the system generates anomalies for observed modes that are extremely unusual. We have outlined a procedure where the library of exemplar modes can be seeded, and particular modes can be labeled as to whether or not they warrant an alert. In this way, an attacker cannot train the system to accept frequently seen malicious behavior as normal. Such behavior instead reinforces a known malicious mode.

At present, the system is integrated into the EMERALD Bayes TCP sensor. We are exploring methods whereby a Bayes-like framework can be used as the basis of a hybrid system incorporating Bayes, anomaly, and signature detection techniques.

References

[Agr88] Agrawal, R., Imielinski, T., and Swami, A. "Mining Association Rules Between Sets of Items in Large Databases," Proceedings of the 1993 ACM SIGMOD Conference.

[Ch96] Cheeseman, P. "Bayesian Classification (Autoclass): Theory and Results", in Fayyad, U., Piatesky-Shapiro, G., and Uthurusamy, R., eds., "Advances in Knowledge Discovery and Data Mining". AAAI Press/MIT Press, 1996.

[De77] Dempster, A P., Laird, N. M., and Rubin, D. B. "Maximum Likelihood from Incomplete Data Sets by the EM Algorithm", Journal of the Royal Statistical Society 39(1): pp. 1-38, 1977.

[For96] Forest, S., Hofmeyr, S., Somayaji, A., and Longstaff, T. "A Sense of Self for Unix Processes," proceedings of the IEEE Symposium on Security and Privacy, Berkeley, CA, May 1996.

[Es02] Eskin, E., Arnold, A., Portnoy, L., and Stolfo, S. "A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data", http://www.cs.columbia.edu/~eeskin/

[Gro88] Grossberg, S. (ed.) "Neural Networks and Natural Intelligence," MIT Press, 1988.

[Jav91] Javitz, H. and Valdes, A. "The SRI IDES Statistical Anomaly Detector," Proceedings of the IEEE Symposium in Security and Privacy, Oakland, CA, May 1991.

[Lee99] Lee, W., Stolfo, S., and Mok, K. "A Data Mining Framework for Building Intrusion Detection Models," 1999 IEEE Symposium on Security and Privacy.

[Lip00] Lippmann, Richard, et al. "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-Line Intrusion Detection Evaluation," Proceedings of DARPA Information Survivability Conference and Exposition, DISCEX'00, Jan 25-27, Hilton Head, SC, 2000, http://www.ll.mit.edu/IST/ideval/index.html

[Max02] Maxion, R. and Tan, K. "Why 6? Defining the Operational Limits of STIDE, and Anomaly-Based Intrusion Detector," Proceedings of the 2002 IEEE Symposium on Security and Privacy, Berkeley, CA, May 2002.

[McH01] McHugh, John, discussion at RAID 2001, Davis, CA, October 2001.

[Rum88] Rummelhart, D. and Zipser, D. "Feature Discovery by Competitive Learning," in Rummelhart, D. and McLelland, J., eds., "Parallel Distributed Processing," MIT Press, 1988

[SD02] Silicon Defense. Statistical Portscan Intrusion Correlation Engine/Statistical Port Anomaly Detector. http://www.silicondefense.com/software/spice/index.htm

[Tou74] Tou, J. T., and Gonzalez, R. C. "Pattern Recognition Principles," Addison-Wesley, 1974.

[Va00] Valdes, A. and Skinner, K. "Adaptive, Model-based Monitoring for Cyber Attack Detection," proceedings of "Recent Advances in Intrusion Detection (RAID 2000)," Toulouse, France, October 2000.

[War99] Warrender, C., Forest, S., and Pearlmutter, B. "Detecting Intrusions Using System Calls: Alternative Data Models," proceedings of the IEEE Symposium on Security and Privacy, Berkeley, CA, May 1999.

[Wes00] Wespi, A., Dacier, M., and Debar, H. "Intrusion Detection Using Variable-Length Audit Trail Patterns," Proceedings of "Recent Advances in Intrusion Detection (RAID 2000)," Toulouse, France, October 2000.

Appendix: Special Nodes in Hybrid Systems

Although in general Bayes networks deal in conditional probability relationships that are learned or approximated from data, it is possible to include in a system certain nodes that establish or refute one or more hypotheses. These are especially useful for classes of evidence that are rarely observed (so that learning conditional probabilities from data will not be reliable) but for which judgment is appropriate. For these nodes, learning should be disabled.

The nodes in question have two states (yes or no, depending on whether the corresponding evidence is observed or fact is established). We will call such nodes establishing or refuting nodes, since they either establish or refute a hypothesis, respectively. The parent node of establishing or refuting nodes can have a number of hypotheses. The framework is presented in a way that incorporates either requirement, since to establish a hypothesis (or subset of hypotheses) is to refute all others. The goal is to achieve the desired establishment or refutation if the evidence is observed (that is, if the node in question is yes) but change belief minimally otherwise. We shall show that nodes of this type are an effective way of incorporating rule-based subsystems into a hybrid system that performs probabilistic inference as well.

Refuted hypotheses are zeroed out or severely downgraded if the evidence is observed (or equivalently, if an appropriate rule fires). Remaining hypotheses increase in posterior belief in proportion to their belief before the observation of the evidence. For example, suppose that hypotheses A, B, and C have beliefs 0.25, 0.50, and 0.25, and evidence is observed that refutes A. Before observing the evidence, B has twice the belief of C. After observation, we would like the belief of A to be near 0, and the belief of B to be approximately twice that of C. In other words, we would like the new beliefs to be near 0.0, 0.67, and 0.33.

A Bayes net made up entirely of nodes of these types can accomplish much of the functionality of a rule-based system. The antecedent of a rule is analogous to an establishing node, and observing "yes" is equivalent to asserting the antecedent. The consequent is the parent node, or more specifically the set of hypotheses that result from the antecedent. With a multilevel Bayes structure, these nodes may represent directly observable evidence or may themselves be parent nodes, so that their assertion is not the result of direct observation but of inference.

Since these nodes are "handcrafted," the developer is cautioned that it is easy to build a system with such nodes that may cause a contradiction, so these nodes must be employed judiciously.

In the following, we assume that α and ε are both small, and ε is small relative to α. Setting ε to 0 causes nodes of this type to make "hard" calls. This can cause numerical problems if, for example, two nodes conflict. The conditional probability table (CPT) for an establishing node is given below. Columns correspond to states of the node in question (yes or no, with yes corresponding to the first column), while rows correspond to hypotheses at the parent node.

$$CPT = \begin{array}{c} H_1 \\ \vdots \\ H_i \\ \vdots \\ H_n \end{array} \begin{bmatrix} \varepsilon & 1-\varepsilon \\ \vdots & \vdots \\ \alpha & 1-\alpha \\ \vdots & \vdots \\ \varepsilon & 1-\varepsilon \end{bmatrix}$$

Here, hypotheses indexed by i are established by the node, and their likelihood is increased by the node according to the ratio α/ε. If the "no" state is observed, the likelihood is slightly diminished for these hypotheses by the ratio $1-\alpha/1-\varepsilon$, which is near unity by construction.

Numerical Example

We now give a numerical example of a node that establishes hypotheses B and C (or equivalently, refutes A). We have chosen $\alpha = 0.05$ and $\varepsilon = 0.01$.

$$CPT = \begin{array}{c} A \\ B \\ C \end{array} \begin{bmatrix} 0.01 & 0.99 \\ 0.05 & 0.95 \\ 0.05 & 0.95 \end{bmatrix}$$

Let us suppose that the belief vector before considering this node is given by

$$Bel[A \quad B \quad C] = [0.25 \quad 0.50 \quad 0.25]$$

We would like the node to refute (significantly downweight) A while leaving B twice as likely as C.

The node computation can be summarized as follows. First, we form the matrix product of the CPT with the likelihood vector corresponding to the observed evidence. If we observe "yes," this vector is given by

$$\lambda = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Since all the likelihood is in one state, this amounts to choosing a column from the CPT. More generally, the likelihood can be fractional corresponding to the observation of "yes" or "no" with some uncertainty. In this case, rather than choosing a column from the CPT, we instead obtain a single column that is a weighted average of the original columns. In either case, we now form the elementwise product of this column with the prior belief and normalize to unit sum. In our case, we obtain

$$Bel = \beta[0.25 \quad 0.50 \quad 0.25] \otimes \begin{bmatrix} 0.01 \\ 0.05 \\ 0.05 \end{bmatrix}$$

$$= \beta[0.0025 \quad 0.025 \quad 0.0125]$$

$$= [0.0625 \quad 0.6250 \quad 0.3125]$$

$$\beta = \text{Normalizing facor, in this case}$$

$$0.0025 + 0.025 + 0.0125$$

We have achieved the desired result of downweighting A and maintaining B twice as likely as C. By choosing α and ε appropriately, we can get as close as we wish to the limiting values

$$Bel = \left[0 \quad {}^{2}\!/_{3} \quad {}^{1}\!/_{3}\right]$$

Now, suppose the node state is "no". We then choose the second column of the CPT. Proceeding as above, it is straightforward to show that the posterior belief becomes

$$Bel = [0.2578 \quad 0.4948 \quad 0.2474]$$

As desired, this represents a minimal change from the prior belief. By increasing α, the node downgrades hypotheses accordingly when the node state is "no"; by setting α to unity, the node can refute these hypotheses.