Artificial Intelligence and Intrusion Detection: Current and Future Directions

Jeremy Frank
frank@cs.ucdavis.edu
Division of Computer Science
University of California at Davis
Davis, CA. 95616
(916) 752-2149
NSA URP MDA904-93-C-4085
ARPA DOD/DABT63-93-C-0045

June 9, 1994

Abstract

Intrusion Detection systems (IDSs) have previously been built by hand. These systems have difficulty successfully classifying intruders, and require a significant amount of computational overhead making it difficult to create robust real-time IDS systems. Artificial Intelligence techniques can reduce the human effort required to build these systems and can improve their performance. Learning and induction are used to improve the performance of search problems, while clustering has been used for data analysis and reduction. AI has recently been used in Intrusion Detection (ID) for anomaly detection, data reduction and induction, or discovery, of rules explaining audit data. We survey uses of artificial intelligence methods in ID, and present an example using feature selection to improve the classification of network connections. The network connection classification problem is related to ID since intruders can create "private" communications services undetectable by normal means. We also explore some areas where AI techniques may further improve IDSs.
Keywords: Artificial Intelligence, Intrusion Detection, Feature Selection.

1 Problems in Intrusion Detection

Intrusion Detection (ID) is the identification of attempted or ongoing attacks on a computer system or network. Issues in ID research include data collection, data reduction, behavior classification, reporting and response. Although there are many significant open problems in ID research, we focus on data reduction and classification. *Data reduction* consists of analyzing a collection of data in order to identify the most important components of the data, thereby reducing processing time, communications overhead and storage requirements. *Classification* is the process of identifying attackers and intruders. Artificial intelligence (AI) techniques have been used in many IDSs to perform these important tasks.

Section 2 of this paper will briefly discuss artificial intelligence methods and describe some of the methods which will appear in this paper. Section 3 will discuss the problem of data reduction and discuss how AI methods have been used in a variety of IDSs to solve this problem. Section 4 will discuss the application of AI to the classification problem. Section 5 will present an example of the use of feature selection to improve the classification of network connections, and section 6 will discuss some future applications of AI in IDSs.

2 Artificial Intelligence Methods

Artificial Intelligence is concerned with improving algorithms by employing problem solving techniques used by human beings. Humans excel at tasks such as *learning*, or gaining the ability to perform tasks from examples and training. An *expert system* handles problems using a computer model of expert human reasoning. However, most expert systems must undergo continuous maintenance to perform well [WeKu].

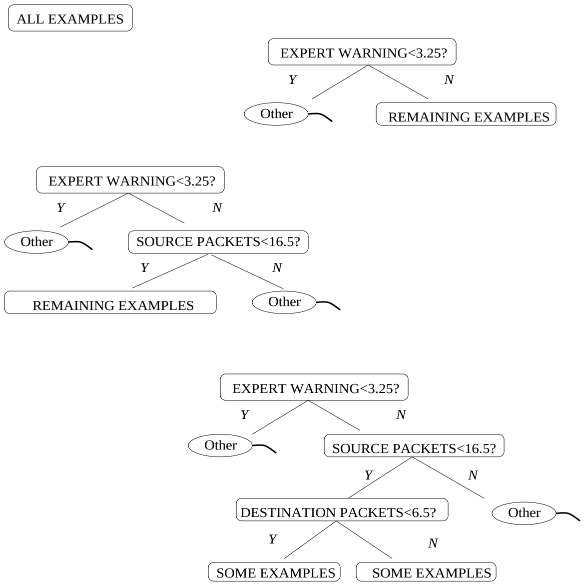

Figure 1: Growing a Decision Tree.

Other systems can acquire knowledge from a set of *training instances*. These training instances can be questions and correct answer pairs, or problems and the steps of a solution. *Rule Based Induction* derives rules which explain the training instances more clearly than a mathematical or statistical analysis of data [WeKu]. *Classifier systems* attempt to learn how to classify future examples from a set of training data. An example of a system that can be used as a classifier is a *Neural Network*, which uses a model of biological systems to perform classification. Neural networks are characterized by highly connected networks which are trained on a set of data in the hopes that the network will correctly classify future examples [WeKu]. Another example of a classifier is a *Decision Tree* [BuCa] [WeKu]. Decision trees are constructed by finding ways to separate the data into two or more groups. We then separate each of these groups in turn, until we have small groups of examples left. Decision tree algorithms are designed to find the best questions to ask so that most or all of the examples in each group belong to one class. Figure 1 shows how a tree used to classify network connections is constructed. The goal of *Feature Selection* is to reduce the amount of information required to make good predictions, and to improve the error rate of classifiers [WeKu]. This is accomplished by searching subsets of features, or information sources, and testing the ability of those features to classify the training instances. The search process itself is the subject of continuing research in the AI community. Humans are also able to generalize or abstract from large amounts of information by a process called *discovery* or *clustering. Data clustering* techniques are used to group data together according to some criteria [ShDi]. Clustering is used to discover hidden patterns in data that humans might miss.

3 Data Reduction for Intrusion Detection

Due to the massive amount of audit data available, classification by hand is impossible. For example, a user typically generates between 3-35Mbytes of data in an eight hour period and it can take hours to analyze a single hour's worth of data. Analysis is difficult even with computer assistance because extraneous features can make it harder to detect suspicious behavior patterns. Complex relationships exist between the features which are difficult for humans to discover. IDSs must therefore reduce the amount of data to be processed. This is especially important if real-time detection is desired. Therefore, some form of data reduction is required for IDSs. Reduction can occur in one of several ways. Data that is not considered useful can be filtered, leaving only the potentially interesting data. Data can be grouped or clustered to reveal hidden patterns; by storing the characteristics of the clusters instead of the data, overhead can be reduced. Finally, some data sources can be eliminated using feature selection.

3.1 Data Filtering

The purpose behind data filtering is to reduce the amount of data directly handled by the IDS. Some data may not be useful to the IDS and thus can be eliminated before processing. This has the advantage of decreasing storage requirements and reducing processing time. However, filtering may throw out useful data, and so must be done carefully.

In systems such as DIDS [SnBr], MIDAS [SeSh], TIM [TeCh] and NSM [HeDi], data filtering is done using heuristic or ad hoc methods, which can be viewed as expert rules for filtering. Other systems filter data in a more adaptive or dynamic way. [DeBe] present a filtering system based on a neural network which acts to filter data which does not fit an observed trend. They assume that user activity contains notable trends that can be detected, and that there are correlations among the collected audit data. Regularity ensures that the network will pick up the regular trends exhibited, and automatically account for correlations in the input data. Using a type of neural network called a *recurrent network* ensures that behavior trends can be accurately recalled. The network "forgets" behavior over time, and can thus adjust to new trends. Thus the network acts as a filter to determine whether or not an audit record fits the regular trends.

3.2 Feature Selection

In complex classification domains, some data may hinder the classification process. Features may contain false correlations which hinder the process of detecting intrusions. Further, some features may be redundant since the information they add is contained in other features. Extra features can increase computation time, and can impact the accuracy of an IDS. Feature selection improves classification by searching for the subset of features which best classifies the training data [SiSk] . In the ID domain, features are derived from information sources used to detect intrusions, and training instances are derived from detected intrusion attempts as well as normal behavior. Thus, feature selection can be used to find features most indicative of misuse, or can be used to distinguish between types of misuse. [Do] and [SiSk] have performed comparisons of a variety of feature selection techniques, and [Do] tested several techniques on simulated computer attack data to explore the possibility of using feature selection to improve intrusion detection techniques. In section 5 we give an example of feature selection applied to classifying network connections.

3.3 Data Clustering

Clustering can be performed to find hidden patterns in data and significant features for use in detection. Clustering can also be used as a reduction technique by storing the characteristics of the clusters instead of the actual data. Artificial Intelligence researchers have noted the close relationship between learning and data compression. Discovering the generalization of a concept is in essence finding a more compact representation of the set of objects, and a hierarchical clustering algorithm can be used for inductive generalization [Th]. Statistical clustering measures the probability that each example is in a given cluster. Exemplar methods build a representative of each cluster throughout the clustering process. Distance clustering uses a distance measure to establish membership in a cluster. Conceptual clustering requires that an object meet necessary and sufficient conditions for cluster membership.

PRAD [LaBe] uses k-nearest neighbor (knn) clustering to reduce data. To perform knn clustering, x percentiles of the distribution are determined. The data is reduced to one of the values 1 to x. Thus each of n data elements is clustered with $k = \frac{n}{x+1}$ neighboring data points. Along with the choice of number of percentile points, the positions of the percentiles can also be located. For instance, [La] uses 2 percentiles and splits the categories at the 50th percentile. The Bernoulli vector used in Haystack uses $x = 2$ and splits at the 90th percentile. [He]

Wisdom&Sense [LiVa] also performs clustering of numerical data. The history of audit data is separated into clusters which correspond to high density regions followed by low density regions; the historical data is then represented by clusters which represent each density region.

4 Behavior Classification in Intrusion Detection

Classifying user or system behavior is a very hard problem. One problem is that only a small fraction of behavior is misuse; another is that often misuse looks like normal use, so it can be difficult to distinguish between intruders and normal users. As a result, classification can result in "false negatives", wherein an attacker is misclassified as a normal user. "False positives" can also degrade productivity in the systems being protected by invoking countermeasures unnecessarily. Finally, all types of intrusive behavior can't be identified in advance. Several AI techniques have been used to improve IDS classification performance. Statistical anomaly detection works on the assumption that many attackers behave differently from normal users, or that a system or process behaves differently during an attack. If a user is behaving abnormally it may indicate an attacker using that user's account. Expert systems encode policy statements and known attacks as a fixed set of rules. User behavior is matched to these rules to determine if an attack is under way. Rule-based systems create (discover) and manage rules corresponding to anomalous behavior.

4.1 Expert Systems

In an expert system, a set of rules encoding knowledge of an "expert" are used to make conclusions about information gathered by the IDS. The rule set must be modified by hand, and may incorporate a statistical or probabilistic component. However, in specialized domains expert systems can outperform humans.

IDES contains a rule-based component which encodes knowledge about past intrusions, known system vulnerabilities, and security policy. IDES rules are encoded in an expert system shell. As information is gathered, the expert determines whether or not any rules have been satisfied, then chooses the most appropriate rule to select [LuJa]. [DeBe] propose an expert system in connection with a neural network. The neural network component reports anomalies to the expert system, which also employs data not used by the net. The expert contains a rule base similar to that used in IDES, with known attacks and system policy information. It also provides the network with contextual inputs that audit data does not provide, and ensures that the network does not train on intrusive behavior. Other IDSs employing expert systems are Haystack [He], AudES [Ts], MIDAS [SeSh] and NADIR [JaDu].

4.2 Anomaly Detection

Anomaly detection is based on the assumption that misuse or intrusive behavior deviates from normal system use [LuTa] [De] [DeBe] [LiVa]. In many cases this is a valid assumption, as in the attacker who breaks into a legitimate user's account. The attacker may behave differently than the regular user, so if the IDS has established what the user normally does during a session, it can determine that the user is not behaving normally and detect the attack. IDSs constructed with this philosophy learn profiles of behavior and report anomalies to either a human or another part of the IDS for more detailed analysis. An anomaly detection system contains three distinct phases:

1. Abstract local information
2. Evolve background information from local abstractions
3. Establish anomaly background boundaries.

[Ma] discusses smoothing raw data to eliminate reliance on outlying data points, blending data using an exponential method to weight historical data higher than current data, and finding and blending the variation in behavior to establish a tolerance level for network anomaly detection. Anomaly detection can be difficult since the concept of normal can change over time. Furthermore, normalcy can be established with respect to different time frames. For example, a system can establish session, daily and weekly trends.

PRAD [La] learns profiles of resource usage, time information, and directory access patterns. Profiles are analyzed with respect to login sessions and time windows, and performance in windows is weighted over time. Windows extend across logins so that information across login sessions can be maintained in the profiles. PRAD provides a means for including changing legitimate user behavior in profiles after legitimacy is ascertained. Other systems employing statistical anomaly detection are MIDAS[SeSh] , IDES [JaVa], NADIR [JaDu] and Haystack [He].

4.3 Rule-Based Induction

In contrast to expert systems, rule-based systems automatically develop rules to explain the historical data they collect. Rules are modified over the lifetime of a system in order to keep the rule set accurate and manageable.

In Wisdom&Sense, rules are generated which specify legal values of features conditioned on the values of other features. Legality is determined from the history of data for each feature. Rules can overlap in specificity due to incomplete information in the history. Rule pruning occurs if there are too many legal values for a feature, too few historical values, the rule is too deep, if rules overlap, or a rule is conditioned on a previously (in the forest of rules) determined anomalous value. All rules can either be used to signal anomalies, or the most appropriate rules to use may be determined [LiVa]. TIM's rules remain in the rule base only if they are highly predictive or confirmed by many observations. Prediction is calculated using an entropy model. The user must specify the behavior TIM is trying to predict. Rules are stored in a lattice, and predict an outcome with a specific accuracy based on the observed audit history. Both short term and long term patterns are checked against for anomalies. Rules also support instantiation. TIM allows the user to enter rules describing either patterns or abstractions of the audit data. [TeCh]

5 An Experiment Using Feature Selection

A growing problem in intrusion detection is network-based intrusion detection. Since computer systems are increasingly network dependent it is imperative to protect both local and regional networks. An example of the kinds of problems that must be faced can be seen in the problem of classifying a network connection. On UNIX systems a connection is characterized by the source port and destination port numbers. Certain ports are reserved for different services; e.g. telnet uses port 23. However, an intruder can hide network connections by strategically placing the servers that receive the connections on different ports[He2]. The mapping of ports to services is internal to a single machine; an intruder could also change the port map. Thus we would like to be able to identify the type of connection made without referring to port numbers.

We examined how feature selection can improve classification of network connections by minimizing the classification error rate and by reducing the number of features required to classify connections. To do so, we analyzed three feature selection algorithms to test methods for selecting the best subset of features to classify connections using decision trees. We conducted two types of experiments: one selecting features which distinguish one type of connection from all others, and one which classified all connection types. Our data consisted of 15,947 connections from one local area network during one week of normal use.

5.1 NSM Features

We collected information about network connections using NSM [HeDi]. NSM returns data about each connection. We collected the following information for each connection:

Feature	Feature type
Index	int
Expert system warning	float
Time in seconds	int
# of packets from source	int
# of packets from destination	int
# of data bytes from source	int
# of data bytes from destination	int

Most of the fields are self-explanatory; the expert system warning value is NSM's expert analysis of how likely the connection is to be an attack. Each piece of information collected was used as a feature for our experiments. In addition to these fields we also collected the actual connection type for use in training.

5.2 Search Algorithms

To reiterate, feature selection is used to reduce the amount of information required to make good predictions, and to improve the error rate of classifiers. Since our task is to correctly classify future examples based on the training examples, we used the classification error rate of a decision tree to evaluate each set of features [BuCa] [WeKu]. This error rate is computed by counting the misclassifications on a test data set which is independent from the training set [BuCa]. We tested representative algorithms from each of 3 broad classes of search algorithms.

Backward sequential search begins with the full set of features. At each stage of the search, each feature in the remaining set is removed. The best feature to eliminate from the set is determined by comparing the error rates of the classifiers created using the resulting feature sets. Backward Sequential Search runs in polynomial time [Do].

Beam search is a type of best-first search which uses a bounded queue to limit the scope of the search. The queue is ordered from best-state to worst-state, with the best state placed in the front of the queue. The algorithm operates by taking the first state in the queue, the most promising state, and extending the search from that state as in Backward Sequential Search. Each new state visited is placed in the queue in order of the goodness of it's state. If there is no limit on the length of the queue, then Beam Search takes exponential time to complete but if the queue length is 1 then Beam Search takes polynomial time. Thus accuracy can be increased if the queue length is increased [Do].

In Random Generation Plus Sequential Selection, we perform several sequential selections from different places in the search space. As mentioned, the goal is to avoid picking the first good feature set seen on the assumption that other good feature sets are also available. To do so, we generate a random feature set, then perform backward and forward sequential selection on the state. Random Generation runs in polynomial time but is more expensive than Backwards Sequential Selection [Do].

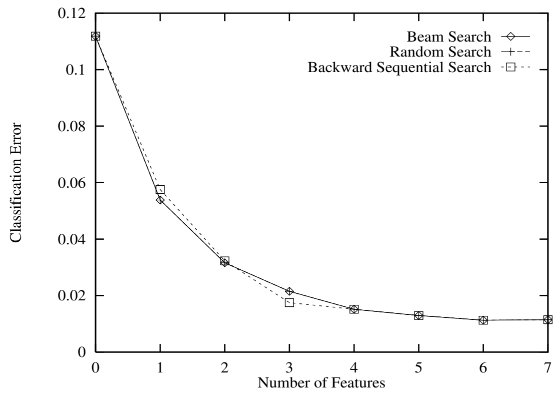

Figure 2: Feature Set Size vs Error Rate For Classifying Connections.

5.3 Algorithm Performance

The table below shows the size, classification error rates, and number of states searched for each problem. Note that with 7 features the total number of subsets of these features is 128. We can see here that in most cases using a computationally expensive algorithm did not gain much in terms of numbers of features or error rate with the exception of the shell classification problem.

Further, the RGSS algorithm always repeated states in it's search. This implies that differentiating connections can be done reasonably well with a less-than-exhaustive search. However, the fact that the random search did find a better classifier indicates that sophisticated algorithms can be worth the cost. Notice that the classification error is smaller than the percentage of the smallest class of connections observed.

Problem	Algorithm	Number of States	Size of Best Feature Set	Error Rate
All	Beam	53	6	0.011266%
All	BSS	29	6	0.011266%
All	RGSS	206	6	0.011266%
SMTP	Beam	53	5	0.007231%
SMTP	BSS	29	5	0.007231%
SMTP	RGSS	190	5	0.007231%
Login	Beam	38	4	0.001177%
Login	BSS	29	4	0.001177%
Login	RGSS	188	4	0.001177%
Shell	Beam	38	4	0.002018%
Shell	BSS	29	4	0.002018%
Shell	RGSS	178	4	0.001009%

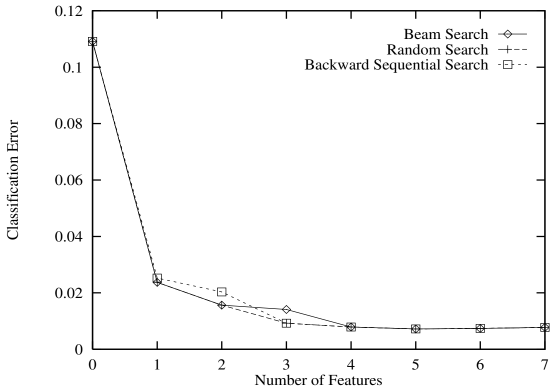

Figure 3: Feature Set Size vs Error Rate For Identifying SMTP Connections.

5.4 Error Rate Performance

The following graphs show how error rate varied with size of feature sets found for each of the problems. We note that in all cases the algorithms performed similarly. We also note that in all cases except the shell connections problem each algorithm found not only the same sized best feature set but that the sets were composed of the same best features.

Figure 2 shows the error rate vs feature set size for classifying all network connections. The error rates are for the best feature set of the indicated size found by each algorithm. We see that with as few as 3 features the error rate is $< 0.02\%$. However, the best feature set only excluded the number of destination data bytes. The 3 features found by all of the search algorithms were time in seconds, packets from the destination and source data bytes, and had an error of 0.017488%. Thus for this particular case a sophisticated algorithm wasn't necessary to find a good classifier.

Figure 3 shows the error rate vs feature set size for classifying SMTP connections. We do not see a jump in the error rate until 3 features, although the best feature set contained 5 features, with the Index and Destination Data bytes excluded. All algorithms found that using time in seconds, packets from the destination and source data bytes gave an error of 0.009248%, so again the less computationally expensive algorithm is the best choice.

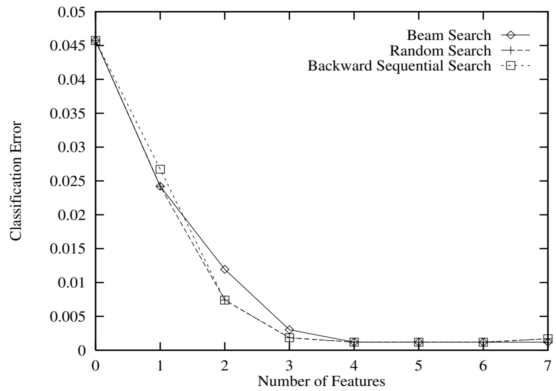

Figure 4: Feature Set Size vs Error Rate For Identifying Login Connections.

Figure 4 shows the error rate vs feature set size for classifying login connections. Again we do not see a jump in the error rate until 3 features, although the best feature set contained 4 features. This time the index, destination data bytes and source data bytes were excluded from the best feature set. Again BSS and RGSS performed the best, with RGSS finding that time in seconds, packets from the destination and source data bytes gave an error of 0.001850%. Again BSS found the same feature set and expanded fewer states.

The results from classifying shell connections were more interesting. While all 3 algorithms found the best feature set size was 4, RGSS found a better feature set with a better error rate, as shown below.

Algorithm	Index	Expert	Time	Src Pkts	Dest Pkts	Src Data	Dest Data
Beam	N	Y	N	Y	Y	Y	N
BSS	N	Y	N	Y	Y	Y	N
RGSS	N	Y	*Y*	Y	*N*	Y	N

Here, all algorithms found sets of 3 features with 5 times the error rate of the best feature set.

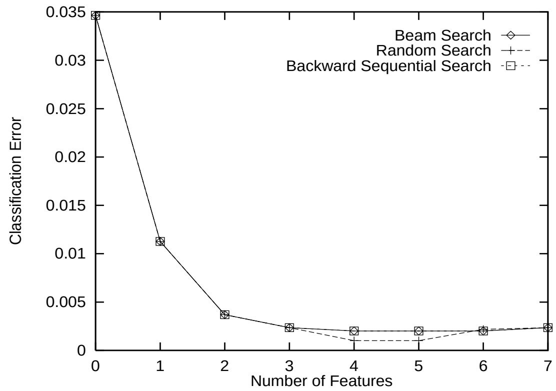

Figure 5: Feature Set Size vs Error Rate For Identifying Shell Connections.

Figure 5 shoes shows the error rate vs feature set size for classifying shell connections.

6 Future uses of AI in Intrusion Detection

Currently, many IDSs employ AI methods in their systems. We expect AI techniques will improve understanding of how non-intrusive and intrusive behavior differ, as well as enable hierarchical classification of different types of attacks.

6.1 Feature Selection in Intrusion Detection Systems

We have demonstrated that feature selection can be effective in a small example. We can extend feature selection to intrusion detection. NSM returns more data than we used in our experiment, and can also be executed containing a list of strings to look for in connections. This information can be invaluable in determining whether critical files or commands pass across the network. However, the number of interesting strings to check for can number easily in the hundreds. Another method of deriving new features is by analyzing multiple transcripts and combining information for a single source-destination pair. Since an attack might span multiple "attempts" over time, a single source-destination pair might appear many times with different data each time. Aggregating statistics and adding others can result in more features. For instance, the number of times a particular connection was made is such an aggregated feature.

6.2 Reconfiguration and Customization of IDSs

IDSs can be site specific. Using data reduction techniques we can customize an IDS to a particular site by finding the information sources most useful to that site's IDS needs. We can also re-configure an IDS using feature selection after finding new data sources. For instance, NSM can be configured to search for different strings on a network. Feature selection can be used to determine which strings are the best to search for.

IDSs such as DIDS, COPS, Haystack and IDES all make assumptions about the type of data they collect. Feature selection techniques can be modified to analyze the value of the features used in other IDSs and perhaps to enhance their performance by eliminating noisy features. Systems like DIDS and IDES may be difficult to analyze, since they incorporate classifier systems already. In some cases the classifiers may rely on all audit features being present before making decisions; if those features are not present it may cause them to incorrectly classify (or fail to classify) behavior.

6.3 Clustering in Intrusion Detection

We envision that clustering will be very useful in intrusion detection. We plan to use clustering techniques such as Autoclass [ChKe] to explore patterns in audit and network data. One way clustering can be used in ID is by giving an overview of complex data. Another is by considering each cluster in turn and analyzing interesting characteristics of each cluster. For instance, if one cluster has characteristics of two other extremely dis-similar clusters, it may indicate usage patterns midway between two "normal" groups, leading to suspicions of misuse.

7 Conclusions

We have provided a brief survey of AI methods used in a variety of IDSs. We have also demonstrated how one technique, feature selection, can be used to reduce overhead and improve classification of network connections. Other IDSs already make extensive use of AI techniques to improve their ability to detect attacks on computer systems.

8 Acknowledgements

Todd Heberlein provided valuable assistance with NSM for this paper. Justin Doak's work provided theoretical basis for this work, and we used his code to provide the data available. He also provided valuable technical assistance. Becky Bace of the NSA also provided valuable comments on the draft of this paper. This work was supported by NSA under University Research Program under Contract Number MDA904-93-C-4085 and by ARPA under Contract Number DOD/DABT63-93-C-0045.

References

[Ba] R. Baldwin. "Kuang: Rule-Based Security Checking." *COPS documentation*, MIT,Lab For Computer Science Programming Systems Research Group, 1989.

[BuCa] W. Buntine, R. Caruna. "Introduction to IND and Recursive Partitioning." *IND Documentation*, NASA Ames Research Center, 1991.

[ChKe] P. Cheeseman, J. Kelley, M. Self, J. Stutz, W. Taylor, D. Freeman. "Autoclass: A Bayesian Classification System. *Proceedings of the 5th International Conference on Machine Learning,* 1988.

[De] D. Denning. "An Intrusion Detection Model." *IEEE Transactions on Software Engineering, vol SE-13, no.2,* 1987.

[DeBe] H. Debar, M Becker, D. Siboni. "A Neural Network Component for an Intrusion Detection System." *Proceedings, IEEE Symposium on Research in Computer Security and Privacy,* 1992.

[Do] J. Doak. "Intrusion Detection: The Application of Feature Selection, A Comparison of Algorithms, and the Application of a Wide Area Network Analyzer." PhD. Thesis, Department of Computer Science, University of California, Davis, 1992

[He] T. Heberlein. "Haystack's Analysis: A Brief Description." Internal Document, University of California, Davis, 1991.

[He2] T. Heberlein, Private Communication, March 1994.

[HeDi] L. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J Wood, D. Wolber. "A Network Security Monitor." *Proceedings, IEEE Symposium on Research in Computer Security and Privacy,* 1990.

[JaDu] K. Jackson, D. DuBois, C. Stallings. "An Expert System Application for Network Intrusion Detection." 1991

[JaVa] H. Javitz, A. Valdes. "The SRI IDES Statistical Anomaly Detector." *Proceedings, IEEE Symposium on Research in Computer Security and Privacy,* 1991.

[LaBe] L. Lankewicz, M. Benard. "Real-time Anomaly Detection Using a Non-Parametric Pattern Recognition Approach." *Proceedings 7th Annual Computer Security Applications Conference,* 1991.

[La] L. Lankewicz. "A Non-Parametric Pattern Recognition to Anomaly Detection." PhD. Thesis, Dept. of Computer Science, Tulane University, 1992.

[LiVa] G. Liepens, H. Vacaro. "Anomaly Detection: Purpose and Framework." *Proceedings, 12th National Computer Security Conference,* 1989.

[LuJa] T. Lunt, R. Jaganathan, R. Lee, A. Whitehurst, S. Listgarten. "Knowledge-Based Intrusion Detection." *Proceedings of the AI Systems in Government Conference,* 1989.

[LuTa] T. Lunt, A. Tamaru, F. Gilham, R. Jaganathan, P. Neuman, C. Jalali. "IDES: A Progress Report." *Proceedings of the Sixth Annual Computer Security Applications Conference,* 1990.

[Ma] R. Maxion, "Anomaly Detection for Diagnosis." *FCTS,* 1990

[RiSc] C. Riesbeck, R. Schank. *Inside Case-Based Reasoning.* Lawrence Erlbaum Associates, New Jersey, 1989.

[SeSh] M. Sebring, E. Shellhouse, M. Hanna, R. Whitehurst. "Expert Systems in Intrusion Detection: A Case Study." *Proceedings of the 11th National Computer Security Conference,* 1988.

[ShDi] J. Shavlik, T. Dietterich. *Readings in Machine Learning.* Morgan Kauffman, California, 1990.

[SiSk] W. Siedlecki, J. Sklansky. "On Automatic Feature Selection." *International Journal of Artificial Intelligence, vol.2, no.2,* 1988.

[SnBr] S. Snapp, J. Bretano, G. Dias, T. Goan, L. Heberlein, C. Ho, K. Levitt, B. Mukherjee, S. Smaha, T. Grance, D. Teal, D. Mansur. "DIDS: Motivation, Architecture and an Early Prototype." *Proceedings of the 14th National Computer Security Conference,* 1991.

[TeCh] H. Teng, K. Chen, S. Lu. "Adaptive Real-Time Anomaly Detection Using Inductively Generated Sequential Patterns." *Proceedings, IEEE Symposium on Research in Computer Security and Privacy,* 1990.

[Th] C. Thornton, "A Computational Model for the Data Compression Metaphor." *Proceedings of the 3d International Conference on Artificial Intelligence,* 1990.

[Ts] G. Tsudik. "AudES - An Expert System for Security Auditing." '*Proceedings of AAAI Conference on Innovative Applications in AI,* 1988

[WeKu] S. Weiss, C Kulikowski. *Computer Systems That Learn.* Morgan Kauffman, California, 1991.