Identifying Significant Features for Network Forensic Analysis Using Artificial Intelligent Techniques

Srinivas Mukkamala[1] & Andrew H. Sung[1,2]
[1]Department of Computer Science
[2]Institute for Complex Additive Systems Analysis
New Mexico Tech

Abstract

Network forensics is the study of analyzing network activity in order to discover the source of security policy violations or information assurance breaches. Capturing network activity for forensic analysis is simple in theory, but relatively trivial in practice. Not all the information captured or recorded will be useful for analysis. Identifying key features that reveal information deemed worthy for further intelligent analysis is a problem of great interest to the researchers in the field.

The focus of this paper is the use of artificial intelligent techniques for offline intrusion analysis, to protect the integrity and confidentiality of the information infrastructure. An effective forensic tool is essential for ensuring information assurance by updating the newly identified security breaches in to the organizations protection and detection mechanisms.

Two artificial intelligent techniques are studied: *Artificial Neural Networks* (ANNs) and *Support Vector Machines* (SVMs). We show that SVMs are superior to ANNs for network forensics in three critical respects: 1. SVMs train, and run an order of magnitude faster; 2. SVMs scale much better; and 3. SVMs give higher classification accuracy.

We also address the related issue of ranking the importance of input features, which is a problem of great interest in modeling. Since elimination of the insignificant and/or useless inputs leads to a simplification of the problem and may allow faster and more accurate detection, feature selection is very important in network forensics.

Two methods for feature ranking are presented; the first one is independent of the modeling tool, while the second method is specific to SVMs. The two methods are applied to identify the important features in the 1999 DARPA intrusion data. It is shown that the two methods produce results that are largely consistent.

1. Introduction

This paper concerns network forensics, offline intrusion analysis and the related issue of identifying important input features for computer forensics and intrusion detection. Cyber forensics is a problem of great significance to information infrastructure protection because computer networks are at the core of the operational control of much of the nation's operations. We use two types of learning machines to build network forensic systems: Artificial Neural Networks or ANNs (1) and Support Vector Machines or SVMs (2). Since the ability to identify the important inputs and redundant inputs of a classifier leads directly to reduced size, faster training and possibly more accurate results, it is critical to be able to identify the important features of network traffic data for cyber forensics in order for the analyzing systems to achieve maximal performance. Therefore, we also study feature ranking and selection, which is itself a problem of great interest in building models based on experimental data.

Ware and Steven Levy pointed out the need for computer security in the early 80's [1,2]. Since most of the intrusions can be located by examining patterns of user activities and audit records [3], many computer forensic tools have been built by utilizing the recognized attack and misuse patterns, which requires human intervention. In our recent work on offline intrusion analysis, artificial intelligence techniques are developed to automate the process by reducing human intervention. SVMs are found to be superior to ANNs in many important respects of intrusion detection [4,5]. In this paper we will concentrate on SVMs and briefly summarize the results of ANNs.

The data we used in our experiments originated from MIT's Lincoln Labs. It was developed for offline intrusion detection system evaluations by DARPA and is considered a benchmark for intrusion detection evaluations [6].

We performed experiments to rank the importance of input features for each of the five classes (normal, probe, denial of service, user to super-user, and remote to local) of patterns in the DARPA data. It is shown that using only the important features for classification provides accuracy and, in certain cases, reduces the training time and testing time of the SVM classifier.

A brief introduction to the data we used is given in section 2. In section 3 we describe the method of deleting one input feature at a time and the performance metrics considered for deciding the importance of a particular feature. In section 4 we present the experimental results of using SVMs for feature ranking. In section 5 we present the experimental results of using ANNs for feature ranking. In section 6 we summarize our results.

1.1 Network Forensics

Network forensics is the act of capturing, recording, and analyzing network audit trails in order to discover the source of security breaches or other information assurance problems. The term network forensics was introduced by the computer security expert Marcus Ranum in the early 90's [7], and is borrowed from the legal and criminology fields where "forensic*s*" pertains to the investigation of crimes. According to Simson Garfinkel, network forensic systems can be implemented in two ways: "catch it as you can" and "stop look and listen" systems [8].

Most of the current techniques are passive monitors, which heavily rely on network traffic, CPU usage, or I/O usage with human intervention. In most of the cases new attacks signatures are detected manually or in some cases go undetected until the incident is being reported. The main focus in the area of cyber forensics is to automate the process of detecting all the attacks and to prevent the damaged caused by further security breaches. The main idea of network forensics is to identify all possible security violations and build these signatures into the detection and prevention mechanisms to prevent further losses.

1.2 Network Forensic Systems

Network forensic systems are designed to identify unauthorized use, misuse, and attacks on information systems which were previously identified manually or in most cases, unidentified. Network forensic systems are mainly classified as “Catch it as you can” and “Stop look and listen” systems. Both of these systems rely on events that took in the past (audit based).

Most network forensic systems are based on audit trails. Systems relying on audit trails try to detect known attack patterns, deviations from normal behavior, or security policy violations. They also try to reduce large volumes of audit data to small volumes for interesting data. One of the main problems with these systems is the overhead, which can become unacceptably high. To analyze logs, the system must keep information regarding all the actions performed, which invariably results in huge amounts of data, requiring disk space and CPU resources. Next, the logs must be processed to convert them into a manageable format, and then compared with the set of recognized misuse and attack patterns to identify possible security violations. Further, the stored patterns need to be continually updated, which would normally involve human expertise. An intelligent, adaptable and cost-effective tool that is capable of this is the goal of the researchers in cyber forensics.

2. The Data

In the 1998 DARPA intrusion detection evaluation program, an environment was set up to acquire raw TCP/IP dump data for a network by simulating a typical U.S. Air Force LAN. The LAN was operated as a real environment, but one that was being blasted with multiple attacks. For each TCP/IP connection, 41 various quantitative and qualitative features were extracted [9]. From this database a subset of 494,021 data were used, of which 20% represent normal patterns.

Attack types fall into four main categories:

1. Probing: surveillance and other probing
2. DOS: denial of service
3. U2Su: unauthorized access to local super user (root) privilege
4. R2L: unauthorized access from a remote machine

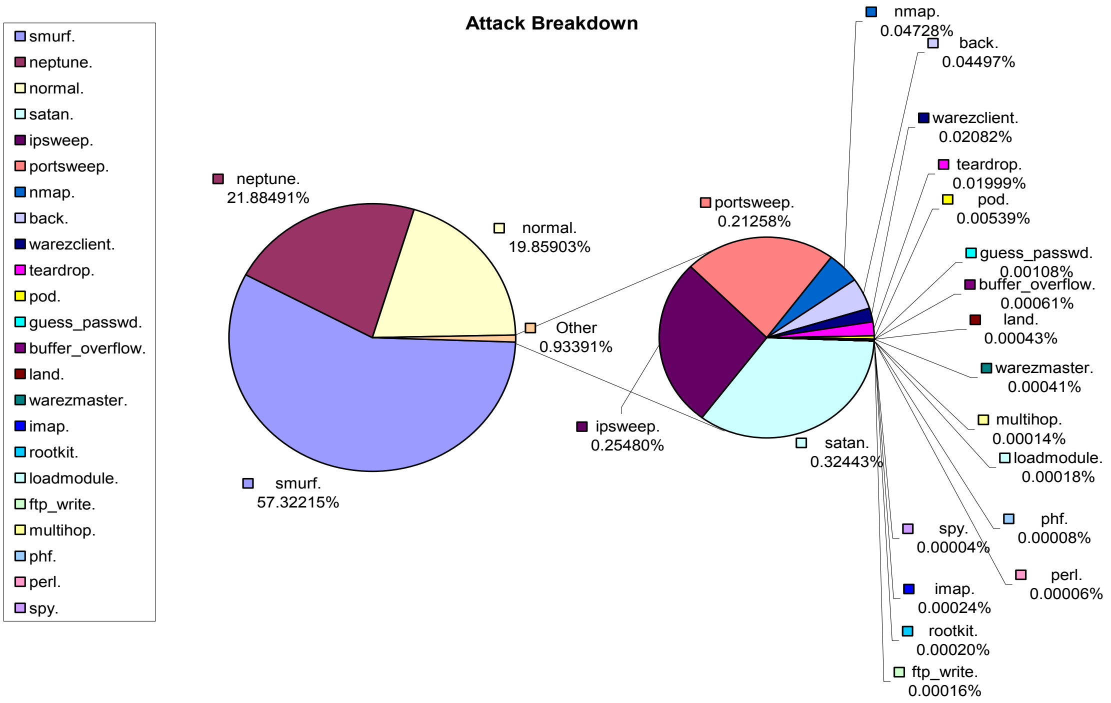

Figure1: Data Distribution

Probing:
Probing is a class of attacks in which an attacker scans a network of computers to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use this information to look for exploits. Examples are Ipsweep, Mscan, Nmap, Saint, Satan

Denial of Service Attacks:
A denial of service attack is a class of attacks in which an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine. Examples are Apache2, Back, Land, Mail bomb, SYN Flood, Ping of death, Process table, Smurf, Syslogd, Teardrop, Udpstorm.

User to Root Attacks:
User to root exploits are a class of attacks in which an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system. Examples are Eject, Ffbconfig, Fdformat, Loadmodule, Perl, Ps, Xterm.

Remote to Local Attacks:
A remote to local attack is a class of attacks in which an attacker who does not have an account exploits some vulnerability to gain local access and sends packets to the machine over a network. Examples are Dictionary, Ftp_write, Guest, Imap, Named, Phf, Sendmail, Xlock, Xsnoop.

3. Ranking The Significance Of Inputs

Feature selection and ranking [10,11] is an important issue in network forensics. Of the large number of features that can be monitored for cyber forensics purpose, which are truly useful, which are less significant, and which may be useless? The question is relevant because the elimination of useless features (or audit trail reduction) enhances the accuracy of detection while speeding up the computation, thus improving the overall performance of the detection mechanism. In cases where there are no useless features, concentrating on the most important ones may well improve the time performance of the detection mechanism, without affecting the accuracy of detection in statistically significant ways.

The feature ranking and selection problem for cyber forensics is similar in nature to various engineering problems that are characterized by:

- Having a large number of input variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ of varying degrees of importance; i.e., some elements of **x** are essential, some are less important, some of them may not be mutually independent, and some may be useless or noise.
- Lacking an analytical model or mathematical formula that precisely describes the input-output relationship, $\mathbf{y} = \boldsymbol{F}(\mathbf{x})$.
- Having available a finite set of experimental data, based on which a model (e.g. neural networks) can be built for simulation and prediction purposes.

Due to the lack of an analytical model, one can only seek to determine the relative importance of the input variables through empirical methods. A complete analysis would require the examination of all possibilities, e.g., taking two variables at a time to analyze their dependence or correlation, then taking three at a time, etc. This, however, is both infeasible (requiring 2^n experiments) and not infallible (since the available data may be of poor quality in sampling the whole input space). In the following, therefore, we apply the technique of deleting one feature at a time to rank the input features and identify the most important ones for intrusion detection using SVMs [4,5].

3.1 Performance-Based Method for Ranking Importance

We first describe a general (i.e., independent of the modeling tools being used), performance-based input ranking methodology: One input feature is deleted from the data at a time; the resultant data set is then used for the training and testing of the classifier. Then the classifier's performance is compared to that of the original classifier (based on all features) in terms of relevant performance criteria. Finally, the importance of the feature is ranked according to a set of rules based on the performance comparison.

The procedure is summarized as follows:

```
1. Compose the training set and the testing set;
       for each feature do the following
2. Delete the feature from the (training and testing) data;
3. Use the resultant data set to train the classifier;
4. Analyze the performance of the classifier using the test set, in terms of the selected
   performance criteria;
5. Rank the importance of the feature according to the rules.
```

3.2 Performance Metrics for SVM Based Ranking

To rank the importance of the 41 features in SVM-based IDS, we consider three main performance criteria: overall accuracy of (5-class) classification; training time; and testing time. Each feature will be ranked as "important," "secondary," or "insignificant," according to the following rules that are applied to the result of the performance comparison of the original 41-feature SVM and the 40-feature SVM:

Rule set:

1. ***If*** *accuracy* decreases ***and*** training time increases ***and*** testing time decreases, ***then*** the feature is important.
2. ***If*** *accuracy* decreases ***and*** training time increases ***and*** testing time increases, ***then*** the feature is important.
3. ***If*** *accuracy* decreases ***and*** training time decreases ***and*** testing time increases, ***then*** the feature is important.
4. ***If*** *accuracy* unchanges ***and*** training time increases ***and*** testing time increases, ***then*** the feature is important.
5. ***If*** *accuracy* unchanges ***and*** training time decreases *and* testing time increases, ***then*** the feature is secondary.
6. ***If*** *accuracy* unchanges ***and*** training time increases ***and*** testing time decreases, ***then*** the feature is secondary.
7. ***If*** *accuracy* unchanges ***and*** training time decreases ***and*** testing time decreases, ***then*** the feature is unimportant.
8. ***If*** *accuracy* increases ***and*** training time increases ***and*** testing time decreases, ***then*** the feature is secondary.
9. ***If*** *accuracy* increases ***and*** training time decreases ***and*** testing time increases, ***then*** the feature is secondary.
10. ***If*** *accuracy* increases ***and*** training time decreases ***and*** testing time decreases, ***then*** the feature is unimportant.

According to the above rules, the 41 features are ranked into the 3 types of {Important}, <Secondary>, or (Unimportant), for each of the 5 classes of patterns, as follows:

class 1: {1,3,5,6,8-10,14,15,17,20-23,25-29,33,35,36,38,39,41}, <2,4,7,11,12,16,18,19,24,30,31,34,37,40>, (13,32)

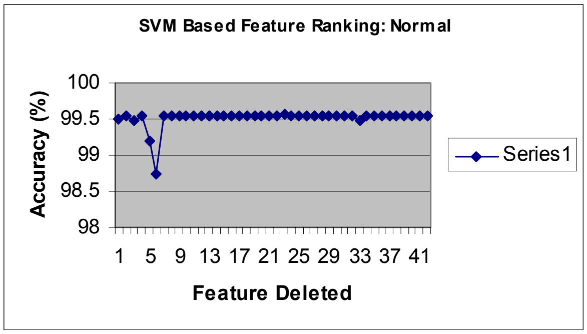

Figure 2: Performance of SVMs using 40 features for Normal

class 2: {3,5,6,23,24,32,33}, <1,4,7-9,12-19,21,22,25-28,34-41>, (2,10,11,20,29,30,31,36,37)

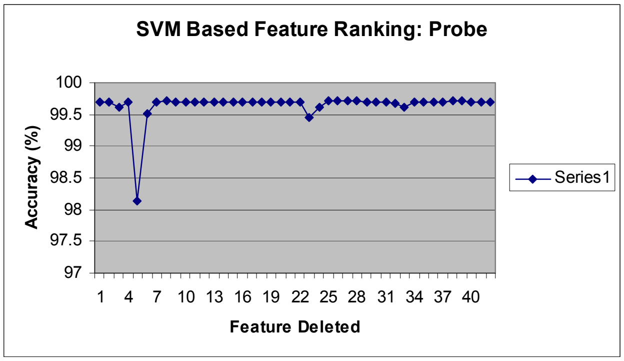

Figure 3: Performance of SVMs using 40 features for Probe

class 3: {1,3,5,6,8,19,23-28,32,33,35,36,38-41}, <2,7,9-11,14,17,20,22,29,30,34,37>, (4,12,13,15,16,18,19,21,3)

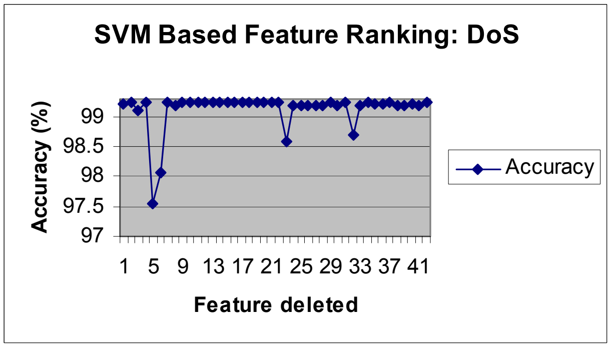

Figure 4: Performance of SVMs using 40 features for Denial of Service

class 4: {5,6,15,16,18,32,33}, <7,8,11,13,17,19-24,26,30,36-39>, (9,10,12,14,27,29,31,34,35,40,41)

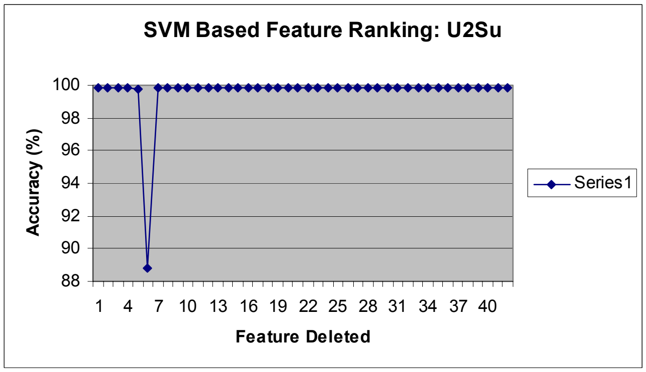

Figure 5: Performance of SVMs using 40 features for User to Root

class 5: {3,5,6,24,32,33}, <2,4,7-23,26-31,34-41>, (1,20,25,38)

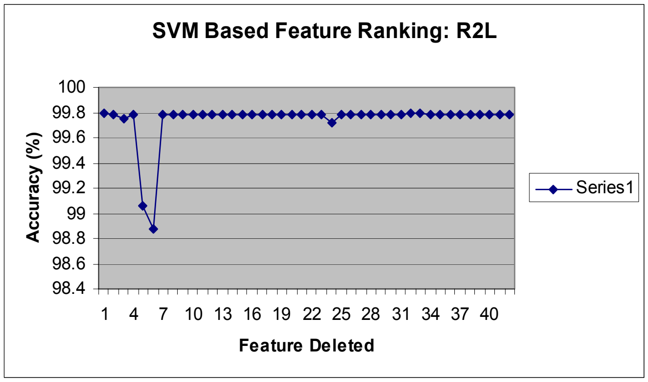

Figure 6: Performance of SVMs using 40 features for Remote to Local

3.3 SVM-specific Feature Ranking Method

Information about the features and their contribution towards classification is hidden in the support vector decision function. Using this information one can rank their significance, i.e., in the equation

$$\mathbf{F\ (X) = \Sigma W_i X_i + b}$$

The point **X** belongs to the positive class if **F**(**X**) is a positive value. The point **X** belongs to the negative class if **F**(**X**) is negative. The value of **F**(**X**) depends on the contribution of each value of **X** and $\mathbf{W}_i$. The absolute value of $\mathbf{W}_i$ measures the strength of the classification. If $\mathbf{W}_i$ is a large positive value then the i^{th} feature is a key factor for positive class. If $\mathbf{W}_i$ is a large negative value then the i^{th} feature is a key factor for negative class. If $\mathbf{W}_i$ is a value close to zero on either the positive or the negative side, then the i^{th} feature does not contribute significantly to the classification. Based on this idea, a ranking can be done by considering the support vector decision function.

3.4 Support Vector Decision Function (SVDF)

The input ranking is done as follows: First the original data set is used for the training of the classifier. Then the classifier's decision function is used to rank the importance of the features. The procedure is:

1. Calculate the weights from the support vector decision function;
2. Rank the importance of the features by the absolute values of the weights.

According to the ranking method, the 41 features are placed into the 3 categories of {Important}, <Secondary> or (Unimportant), for each of the 5 classes of patterns, as follows:

class 1 Normal: {1,2,3,4,5,6,10,12,17,23,24,27,28,29,31,32,33,34,36,39}, <11,13,14,16,19,22,25,26,30,35,37,38,40,41>, (7,8,9,15,18,20,21)

class 2 Probe: {1,2,3,4,5,6,23,24,29,32,33}, <10,12, 22,28, 34,35,36,38,39,41 >, (7,8,9,11,13,14, 15,16, 17,18, 19 20,21,25,26,27,30,31,37,40)

class 3 DOS: {1,5,6,23,24,25,26,32,36,38,39}, <2,3, 4,10, 12,29,33,34 >(7,8,9,11,13,14,15,16, 17 18,19, 20,21, 22,27,28,30,31,35,36,37,40,41) class 4 U2Su: {1,2,3,5,6,12,23,24,32,33}, <4,10,13, 14,17 22,27,29,31,34,36,37,39 > (7,8,9,11,15, 16,18, 19,20, 21,25,26,28,30,35,38,40,41)

class 5 R2L: {1,3,5,6,32,33}, <2,4,10,12,22,23,24, 29,31,34,36,37,38,40 >, (7,8,9,11,13,14,15,16,17, 18, 19,20,21,25,26,27,28,30,35,39,41)

3.5 Performance Metrics for Neural Network Based Ranking

To rank the importance of the 41 features in neural network based model, we consider three main performance criteria: overall accuracy (***OA***) of (5-class) classification; false positive rate (***FP***); and false negative rate (***FN***). Each feature will be ranked as "important," "secondary," or "insignificant," according to the following rules that are applied to the result of performance comparison of the original 41-feature neural network and the 40-feature neural network:
Rule set:

1. *If* ***OA*** increases **and** ***FP*** decreases **and** ***FN*** decreases, then the feature is unimportant.
2. *If* ***OA*** increases **and** ***FP*** increases **and** ***FN*** decreases, then the feature is unimportant.
3. *If* ***OA*** decreases **and** ***FP*** increases **and** ***FN*** increases, then the feature is important.
4. *If* ***OA*** decreases **and** ***FP*** decreases **and** ***FN*** increases, then the feature is important.
5. *If* **OA** un-changes **and** ***FP*** un-changes, then the feature is secondary.

Our performance metrics and the rules for deciding the importance of the input features are not limited to the above-mentioned ones; they can be modified depending on the complexity and the nature of the problem.

According to the ranking method, the 41 features are placed into the 2 categories of {Important} or (Unimportant), for the 5 classes of patterns, as follows.

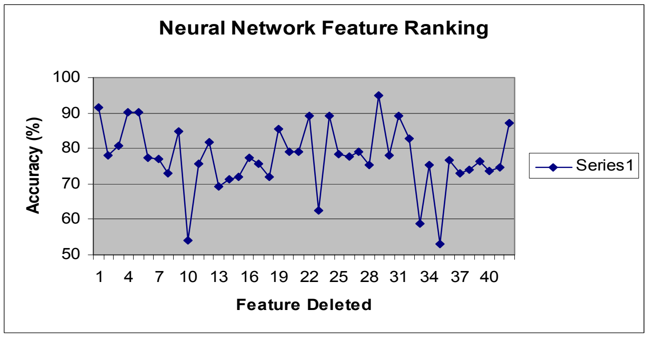

Figure 7: Performance of Neural Networks using 40 features

4. EXPERIMENTS USING Svms

SVMs are used in each of the two methods for ranking the importance of the input features. Once the importance of the input features is ranked, the classifiers are trained and tested with only the important features. Further, we validate the ranking by comparing the performance of the classifier using all input features to that using the important features and we also compare the performance of a classifier using the union of the important features for all fives classes.

(Because SVMs are only capable of binary classifications, we will need to employ five SVMs for the five-class identification problem in intrusion detection. But since the set of important features may differ from class to class, using five SVMs becomes an advantage rather than a hindrance, i.e., in building an IDS using five SVMs, each SVM can use only the important features for that class which it is responsible for making classifications [12].)

4.1 Support Vector Machines

Support vector machines, or SVMs, are learning machines that plot the training vectors in high-dimensional feature space, labeling each vector by its class. SVMs classify data by determining a set of support vectors, which are members of the set of training inputs that outline a hyper plane in the feature space [13,14].

SVMs provide a generic mechanism to fit the surface of the hyper plane to the data through the use of a kernel function. The user may provide a function (e.g., linear, polynomial, or sigmoid) to the SVMs during the training process, which selects support vectors along the surface of this function. The number of free parameters used in the SVMs depends on the margin that separates the data points, but not on the number of input features. Thus SVMs do not require a reduction in the number of features in order to avoid over fitting--an apparent advantage in applications such as intrusion detection. Another primary advantage of SVMs is the low expected probability of generalization errors.

There are other reasons that we use SVMs for intrusion detection. The first is speed; because real-time performance is of primary importance to intrusion detection systems, any classifier that can potentially run “fast” is worth considering. The second reason is scalability; SVMs are relatively insensitive to the number of data points and the classification complexity does not depend on the dimensionality of the feature space, so they can potentially learn a larger set of patterns and scale better than neural networks. Once the data is classified into two classes, a suitable optimizing algorithm can be used, if necessary, for further feature identification, depending on the application.

4.2 SVM Performance Statistics

Our results are summarized in the following tables. Table 1 gives the performance results of the five SVMs for each respective class of data. Table 2 shows the results of SVMs performing classification, with each SVM using as input the important features for all five classes. Table 3 shows the results of SVMs performing classification, with each SVM using as input the union of the important features for all five classes. Table 4 shows the result of SVMs performing classification, with each SVM using as input the important and secondary features for each respective class. Table 5 shows the results of SVMs performing classification, with each SVM using as input the important features obtained from the SVDF ranking. Table 6 shows the results of SVMs performing classification, with each SVM using as input the union of the important features for each class as obtained from the SVDF ranking; the union has 23 features.

Table1: Performance of SVMs using 41 features

Class	Training Time (sec)	Testing Time (sec)	Accuracy (%)
Normal	7.66	1.26	99.55
Probe	49.13	2.10	99.70
DOS	22.87	1.92	99.25
U2Su	3.38	1.05	99.87
R2L	11.54	1.02	99.78

Table2: Performance of SVMs using important features

Class	No of Features	Training Time (sec)	Testing Time (sec)	Accuracy (%)
Normal	25	9.36	1.07	99.59
Probe	7	37.71	1.87	99.38
DOS	19	22.79	1.84	99.22
U2Su	8	2.56	0.85	99.87
R2L	6	8.76	0.73	99.78

Table3: Performance of SVMs using union of important features (30)

Class	Training Time (sec)	Testing Time (sec)	Accuracy (%)
Normal	7.67	1.02	99.51%
Probe	44.38	2.07	99.67%
DOS	18.64	1.41	99.22%
U2Su	3.23	0.98	99.87%
R2L	9.81	1.01	99.78%

Table4: Performance of SVMs using important and secondary features

Class	No of Features	Training Time (sec)	Testing Time (sec)	Accuracy (%)
Normal	39	8.15	1.22	99.59
Probe	32	47.56	2.09	99.65
DOS	32	19.72	2.11	99.25
U2Su	25	2.72	0.92	99.87
R2L	37	8.25	1.25	99.80

Table5: Performance of SVMs using important features as ranked by SVDF

Class	No of Features	Training Time (sec)	Testing Time (sec)	Accuracy (%)
Normal	20	4.58	0.78	99.55
Probe	11	40.56	1.20	99.36
DOS	11	18.93	1.00	99.16
U2Su	10	1.46	0.70	99.87
R2L	6	6.79	0.72	99.72

Table6: Performance of SVMs using union of important features (total 23) as ranked by SVDF

Class	Training time	Testing time	Accuracy (%)
Normal	4.85	0.82	99.55
Probe	36.23	1.40	99.71
DOS	7.77	1.32	99.20
U2Su	1.72	0.75	99.87
R2L	5.91	0.88	99.78

Table7 Performance of SVMs using important and secondary features using SVDF

Class	No of Features	Training Time (sec)	Testing Time (sec)	Accuracy (%)
Normal	34	4.61	0.97	99.55
Probe	21	39.69	1.45	99.56
DOS	19	73.55	1.50	99.56
U2Su	23	1.73	0.79	99.87
R2L	20	5.94	0.91	99.78

5. Experiments Using Neural Networks

This section summarizes the authors' recent work in comparing ANNs and SVMs for intrusion detection [4,5,10,11]. Since a (multi-layer feedforward) ANN is capable of making multi-class classifications, a single ANN (Scaled Conjugate Gradient Decent), is employed to perform the intrusion detection, using the same training and testing sets as those for the SVMs.

Neural networks are used for ranking the importance of the input features, taking training time, testing time, and classification accuracy as the performance measure, and a set of rules is used for ranking. Therefore, the method is an extension of the feature ranking method described in [15] where a cement bonding quality problem is used as the engineering application. Once the importance of the input feature was ranked, the ANNs are trained and tested with the data set containing only the important features. We then compare the performance of the trained classifier against the original ANN trained with data containing all input features.

5.1 Artificial Neural Networks

Artificial neural networks consist of a collection of processing elements that are highly interconnected and transform a set of desired outputs [16,17]. The result of the transformation is determined by the characteristics of the elements and the weights associated with the interconnections among them. A neural network conducts an analysis of the information and provides a probability estimate that it matches with the data it has been trained to recognize. The neural network gains the experience initially by training the system with both the input and outut of the desired problem. The network configuration is refined until satisfactory results are obtained. The neural network gains experience over a period as it is being trained on the data related to the problem.

5.2 ANN Performance Statistics

Table 7 below gives the comparison of the ANN with all 41 features to that of using 34 important features that have been obtained by our feature-ranking algorithm described above.

Table7: Neural network results using all 34 important features

Number of features	Accuracy	False positive rate	False negative rate	Number of epochs
41	87.07	6.66	6.27	412
34	81.57	18.19	0.25	27

6. SUMMARY AND CONCLUSIONS

A number of observations and conclusions are drawn from the results reported:

- SVMs outperform ANNs in the important respects of scalability (SVMs can train with a larger number of patterns, while ANNs would take a long time to train or fail to converge at all when the number of patterns gets large.); training time and running time (SVMs run an order of magnitude faster); and prediction accuracy.
- SVMs easily achieve high detection accuracy (higher than 99%) for each of the 5 classes of data, regardless of whether all 41 features are used, only the important features for each class are used, or the union of all important features for all classes are used.

We note, however, that the difference in accuracy figures tend to be very small and may not be statistically significant, especially in view of the fact that the 5 classes of patterns differ in their sizes tremendously. More definitive conclusions can only be made after analyzing more comprehensive sets of network traffic data.

Regarding feature ranking, we observe that:

- The two feature ranking methods produce largely consistent results. Except for the class 1 (Normal) and class 4 (U2Su) data, the features ranked as Important by the two methods heavily overlap.
- The most important features for the two classes of 'Normal' and 'DOS' heavily overlap.

- ‘U2Su’ and ‘R2L’ are the two smallest classes representing the most serious attacks. Each has a small number of important features and a large number of secondary features.
- The performances of (a) using the important features for each class, Table 2 and Table 5, (b) using the union of important features, Table 3 and Table 6, and (c) using the union of important and secondary features for each class, Table 4 and Table 7, do not show significant differences, and are all similar to that of using all 41 features.
- Using the important features for each class gives the most remarkable performance: the testing time decreases in each class, the accuracy increases slightly for one class “Normal,” decreases slightly for two classes “Probe” and “DOS,” and remains the same for the two most serious attack classes.

Our ongoing experiments include making 23-class (22 specific attacks and normal) feature identifications using SVMs, for the purpose of designing cost-effective and accurate forensic and intrusion detection tools.

7. Acknowledgements

Partial support for this research was received from ICASA (Institute for Complex Additive Systems Analysis, a division of New Mexico Tech), an IASP capacity building grant, and Dr. Peter Gerity (vice president academic affairs, New Mexico Tech). We would also like to acknowledge many insightful suggestions from Dr. Jean-Louis Lassez that helped clarify our ideas and contributed to our work.

8. References

[1] Ware, W. H. “Security Controls for Computer Systems,*” Report of Defense Science Board, Task Force on Computer Security*. Santa Monica, CA: The Rand Corporation, 1979.

[2] Steven Levy, “Hackers - Heroes of the Computer Revolution,” *Dell*, 1984.

[3] Lunt, Teresa F. “A Survey of Intrusion Detection Techniques,” *Computers and Security 12,* 4 (June 1993), pp 405-418.

[4] Mukkamala S., Janoski G., Sung A. H. (2002) “Intrusion Detection Using Neural Networks and Support Vector Machines,” *Proceedings of IEEE International Joint Conference on Neural Networks*, pp.1702-1707.

[5] Srinivas Mukkamala, Andrew H. Sung “Audit Data Reduction Using Neural Networks and Support Vector Machines,” *Digital Forensics Research Workshop* (DFRWS-2002), August 7-9, 2002, Syracuse University, Center for Systems Assurance.

[6] 1998 DARPA Intrusion Detection Evaluation. http://www.ll.mit.edu/IST/ideval/docs/docs_index.html

[7] Marcus Ranum, Network Flight Recorder. http://www.ranum.com/

[8] Simson Garfinkel, Web Security, Privacy & Commerce, 2nd Edition. http://www.oreillynet.com/pub/a/network/2002/04/26/nettap.html

[9] W. Lee and S. Stolfo, “Data Mining Approaches for Intrusion Detection,” *Proc. 1998 7th USENIX Security Symposium*, 1998.

[10] Mukkamala S., Janoski G., Sung A. H. (2001) “Monitoring Information System Security,” *Proceedings of the 11th Annual Workshop on Information Technologies & Systems*, pp.139-144.

[11] S Mukkamala, A H. Sung “Identifying Key Features for Intrusion Detection Using Neural Networks,” *Proceedings of ICCC International Conference on Computer Communications 2002* .

[12] Joachims T. (2000) "SVMlight is an Implementation of Support Vector Machines (SVMs) in C," http://ais.gmd.de/~thorsten/svm_light. University of Dortmund. *Collaborative Research Center on Complexity Reduction in Multivariate Data* (SFB475).

[13] Vladimir V. N. (1995) "The Nature of Statistical Learning Theory," *Springer*.

[14] Joachims T. (2000) "Estimating the Generalization Performance of a SVM Efficiently," *Proceedings of the International Conference on Machine Learning*, Morgan Kaufman.

[15] Sung A. H. (1998) "Ranking Importance of Input Parameters of Neural Networks," *Expert Systems with Applications*, pp.405-41.

[16] Hertz J., Krogh A., Palmer, R. G. (1991) *Introduction to the Theory of Neural Computation, Addison –Wesley.*

[17] Demuth H, Beale M (2000) Neural Network Toolbox User's Guide. Math Works, Inc. Natick, MA.

9. Appendix

Description of a few significant features identified by both ranking methods:

- **Normal**: [1,3,5,6,10,17,23,27,28,29,33,36,39]
  - Duration: Length of the connection made by the destination system to the host system
  - Service: Network service used by the destination system to connect to the host system
  - Source bytes: Number of bytes sent from the host system to the destination system
  - Destination bytes: Number of bytes sent from the destination system to the host system
  - Hot indicators: Number of "hot" indicators
  - File creations: Number of file operations
  - Count: Number of connections made to the same host system in a given interval of time
  - REG error rate: Percentage of connections that have REG error
  - Same service-REG error rate: Percentage of connections with the same service that have REG errors
  - Same service rate: Percentage of connections from the destination system to the host system with the same service in a given interval of time
  - Destination-Host-Service-Count: Number of connections made by the destination system using the same service to the same host system in a given interval of time
  - Destination-Host-Same source- port rate: Percentage of connections from the destination system to the same port of the host system in a given interval of time
  - Destination-Host-Service source- SYN error rate: Percentage of connections from the destination system to the host system with SYN errors in a given interval of time

- **Probe:** [3,5,6,23,24,32,33]
  - Service: Network service used by the destination system to connect to the host system
  - Source bytes: Number of bytes sent from the host system to the destination system
  - Destination bytes: Number of bytes sent from the destination system to the host system
  - Count: Number of connections made to the same host system in a given interval of time
  - Same service rate: Percentage of connections from the destination system to the host system with the same service in a given interval of time

- Destination-Host Count: Number of connections made by the same destination system to the same host system in a given interval of time
- Destination-Host-Same service Count: Number of connections made by the destination system using the same service to the same host system in a given interval of time

- **Denial of Service Attacks**: [1,5,6,23,24,25,26,32,36,38,39]
  - Duration: Length of the connection made by the destination system to the host system
  - Source bytes: Number of bytes sent from the host system to the destination system
  - Destination bytes: Number of bytes sent from the destination system to the host system
  - Count: Number of connections made to the same host system in a given interval of time
  - Same service rate: Percentage of connections from the destination system to the host system with the same service in a given interval of time
  - Connections with SYN errors: Percentage of connections with SYN errors in a given interval of time
  - Connections-Same service-SYN errors: Percentage of connections to the same service with SYN errors in a given interval of time
  - Destination-Host Count: Number of connections made by the same destination system to the same host system in a given interval of time
  - Destination-Host-Same source- port rate: Percentage of connections made by the destination system to the same port on the host system in a given interval of time
  - Destination-Host-SYN error rate: Percentage of connections from the destination system to the host system with SYN errors in a given interval of time
  - Destination-Host-Same-Service error rate: Number of connections made by the destination system using the same service to the same host system in a given interval of time

- **User to Super-User**: [5,6,32,33]
  - Source bytes: Number of bytes sent from the host system to the destination system
  - Destination bytes: Number of bytes sent from the destination system to the host system
  - Destination-Host Count: Number of connections made by the same destination system to the same host system in a given interval of time
  - Destination-Host-Service Count: Number of connections made by the destination system using the same service to the same host system in a given interval of time
- **Remote to Local**: [3,5,6,32,33]
  - Service: Network service used by the destination system to connect to the host system
  - Source bytes: Number of bytes sent from the host system to the destination system
  - Destination bytes: Number of bytes sent from the destination system to the host system
  - Destination-Host Count: Number of connections made by the same destination system to the same host system in a given interval of time
  - Destination-Host-Service Count: Number of connections made by the destination system using the same service to the same host system in a given interval of time

About the Authors

Srinivas Mukkamala (srinivas@ladron.cs.nmt.edu**)** joined the Computer Science graduate program of New Mexico Tech in 2000 and is currently a Ph.D. student. He received his B.E. degree in computer science and engineering in 1999 from University of Madras. His interests are information assurance, information hiding, artificial intelligence, and soft computing techniques for computer security.

Andrew H. Sung (sung@cs.nmt.edu) is professor and chairman of the Computer Science Department, and the Coordinator of the Information Technology Program, at New Mexico Tech. He received his Ph.D. in computer science from the State University of New York at Stony Brook in1984. His interests are intelligent systems, soft computing, and information assurance.