Intrusion Detection Using Neural Networks and Support Vector Machines

Srinivas Mukkamala, Guadalupe Janoski, Andrew Sung
{srinivas, silfalco, sung}@cs.nmt.edu
Department of Computer Science
New Mexico Institute of Mining and Technology
Socorro New Mexico, 87801 USA

Abstract - Information security is an issue of serious global concern. The complexity, accessibility, and openness of the Internet have served to increase the security risk of information systems tremendously. This paper concerns intrusion detection. We describe approaches to intrusion detection using neural networks and support vector machines. The key ideas are to discover useful patterns or features that describe user behavior on a system, and use the set of relevant features to build classifiers that can recognize anomalies and known intrusions, hopefully in real time. Using a set of benchmark data from a KDD (Knowledge Discovery and Data Mining) competition designed by DARPA, we demonstrate that efficient and accurate classifiers can be built to detect intrusions. We compare the performance of neural networks based, and support vector machine based, systems for intrusion detection.

I. Introduction

Information assurance is an issue of serious global concern. The Internet has brought about great benefits to the modern society; meanwhile, the rapidly increasing connectivity and accessibility to the Internet has posed a tremendous security threat. Malicious usage, attacks, and sabotage have been on the rise as more and more computers are put into use. Connecting information systems to networks such as the Internet and public telephone systems further magnifies the potential for exposure through a variety of attack channels.

This paper concerns intrusion detection, an important issue in defensive information warfare. We present the use of neural networks and support vectors machines for intrusion detection of information systems. Since most of the intrusions can be located by examining patterns of user activities, many IDSs have been built by utilizing the recognized attack and misuse patterns. Using neural networks for intrusion detection has been done within the security community [1,7,8,10,11]. In our experiments, the neural networks and support vector machines are trained with normal user activity and attack patterns. The data we used originated from MIT's Lincoln Labs. It was developed for KDD competition by DARPA and is considered a standard benchmark for intrusion detection evaluations. Our goal for intrusion detection is to detect both anomalies and misuses. The approach is to train the neural networks or support vector machines to learn the normal behavior and attack patterns; then significant deviations from normal behavior are flagged as attacks. We begin by giving basic definitions and terms in the next section.

II. Intrusion

Intrusion can be defined as any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource. In the context of information systems, intrusion refers to any unauthorized access, unauthorized attempt to access or damage, or malicious use of information resources. Intrusion can be categorized into two classes, anomaly intrusions and misuse intrusions.

Anomalies are deviations from normal usage behavior. Misuses, on the other hand, are recognized patterns of attack [2]. While misuse patterns are often simpler to process and locate, it is often the anomaly patterns that will help to locate problems. As misuses are recognized patterns of attack, the detection system tends to fail when novel attack methods are implemented. Detection of anomaly patterns is computationally expensive because of the overhead of keeping track of, and possibly updating several system profile metrics, as it must be tailored system to system, and sometimes even user to user, due to the fact behavior patterns and system usage vary greatly.

A. Intrusion Detection

The most popular way to detect intrusions is by using the audit data generated by the operating system. Since almost all activities are logged on a system, it is possible that a manual inspection of these logs would allow intrusions to be detected. It is important to analyze the audit data even after an attack has occurred to determine the extent of damage sustained; this analysis also helps in tracking down the attackers and in recording the attack patterns for future detection. A good IDS that can be used to analyze audit data for such insights makes a valuable tool for information systems.

The idea behind anomaly detection is to establish each user's normal activity profile, and to flag deviations from the established profile as possible intrusion attempts. A main issue concerning misuse detection is how to develop signatures that include all possible attacks to avoid false negatives, and how to develop signatures that do not match non-intrusive activities to avoid false positives. Though false negatives are frequently considered more serious, the selection of threshold levels is important so that neither of the above problems is unreasonably magnified.

0-7803-7278-6/02/$10.00 ©2002 IEEE

B. Intrusion Detection Systems

Intrusion detection systems (IDS) [9] are designed to identify-preferably in real time-unauthorized use, misuse and attacks on information systems. IDSs maintains a set of historical profiles or recorded profiles for users, matches an audit record with appropriate profile, updates the profile whenever necessary, and reports any anomalies detected. An IDS does not usually perform any action to prevent intrusions; its main function is to alert the system administrators that there is a possible security violation; as such it is a proactive tool rather than a reactive tool. IDSs are classified into two types: host based IDS and network based IDS. A host based IDS monitors all the activity on a single information system host. It ensures none of the information system security policies are being violated. A network IDS monitors activities on a whole network and analyzes traffic for potential security breaches or violations.

One of the main problems with IDSs is the overhead, which can become unacceptably high. To analyze system logs, the operating system must keep information regarding all the actions performed, which invariably results in huge amounts of data, requiring disk space and CPU resource. Next, the logs must be processed to convert 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 be continually updated, which would normally involve human expertise. An intelligent, adaptable and cost-effective tool that is capable of (mostly) real-time intrusion detection is the goal.

III. DARPA Data for Intrusion Detection

The data was acquired from the 1998 DARPA intrusion detection evaluation program. They set up an environment to acquire raw TCP/IP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN. They operated the LAN as if it was a true environment, but blasted with multiple attacks. For each TCP/IP connection, 41 various quantitative and qualitative features were extracted.

Attacks fall into four main categories:

1. DOS: denial of service
2. R2L: unauthorized access from a remote machine
3. U2R: unauthorized access to local super user (root) privileges
4. Probing: surveillance and other probing

Table 1 shows 32 different exploits that were used in the 1998 DARPA intrusion detection evaluation. This table presents attacks broken up into categories by type and operating system.

TABLE 1: ATTACKS USED IN DARPA EVALUATION

Attack class	OS: Solaris	OS: SunOS	OS: Linux
Denial of service	Apache2 Back Mail bomb Neptune Ping of death Process table Smurf Syslogd UDP storm	Apache2 Back Mail bomb Neptune Ping of death Process table Smurf Syslogd UDP storm	Apache2 Back Mail bomb Neptune Ping of death Process table Smurf Syslogd UDP storm
Remote to user	Dictionary Ftp-write Guest Phf Xlock Xnsnoop	Dictionary Ftp-write Guest Phf Xlock Xnsnoop	Dictionary Ftp-write Guest Imap Named Phf Sendmail Xlock Xnsnoop
User to super-user	Eject Ffbconfig Fdformat Ps	Load module Ps	Perl Xterm
Probing	Ip sweep Mscan Nmap Saint Satan	Ip sweep Mscan Nmap Saint Satan	Ip sweep Mscan Nmap Saint Satan

A. 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, Mailbomb, SYN Flood, Ping of death, Process table, Smurf, Syslogd, Teardrop, Udpstorm.

B. 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.

C. Remote to User Attacks

A remote to user attack is a class of attacks in which an attacker sends packets to a machine over a network?but who does not have an account on that machine; exploits some vulnerability to gain local access as a user of that machine. Examples are Dictionary, Ftp_write, Guest, Imap, Named, Phf, Sendmail, Xlock, Xsnoop.

D. 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.

E. List of Features

TABLE 2: LIST OF FEATURES
(KDD-CUP-99 TASK DESCRIPTION [16])

Feature name	Description	Type
Duration	Length (number of seconds) of the connection	Continuous
Protocol type	Type of the protocol, e.g. tcp, udp, etc.	Discrete
Service	Network service on the destination, e.g., http, telnet, etc.	Discrete
Src_bytes	Number of data bytes from source to destination	Continuous
Dst_bytes	Number of data bytes from destination to source	Continuous
Flag	Normal or error status of the connection	Discrete
Land	1 if connection is from/to the same host/port; 0 otherwise	Discrete
Wrong_fragment	Number of ``wrong'' fragments	Continuous
Urgent	Number of urgent packets	Continuous
Hot	Number of ``hot'' indicators	Continuous
Num_failed_logins	Number of failed login attempts	Continuous
Logged in	1 if successfully logged in; 0 otherwise	Discrete
Num_compromised	Number of ``compromised'' conditions	Continuous
Root_shell	1 if root shell is obtained; 0 otherwise	Discrete
SU_attempted	1 if ``su root'' command attempted; 0 otherwise	Discrete
Num_root	Number of ``root'' accesses	Continuous
Num_file_creations	Number of file creation operations	Continuous
Num_shells	Number of shell prompts	Continuous
Num_access_files	Number of operations on access control files	continuous
Num_outbound_cmds	Number of outbound commands in an ftp session	Continuous
Is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	Discrete
Is_guest_login	1 if the login is a ``guest' login; 0 otherwise	Discrete
Count	Number of connections to the same host as the current connection in the past two seconds	Continuous
Serror_rate	% Of connections that have ``SYN'' errors	Continuous
Rerror_rate	% Of connections that have ``REJ'' errors	Continuous
Same_srv_rate	% Of connections to the same service	Continuous
Diff_srv_rate	% Of connections to different services	Continuous
Srv_count	Number of connections to the same service as the current connection in the past two seconds	Continuous
Srv_serror_rate	% Of connections that have ``SYN'' errors	Continuous
Srv_rerror_rate	% Of connections that have ``REJ'' errors	Continuous
Srv_diff_host_rate	% Of connections to different hosts	Continuous

IV. SVM Intusion Detection System

The construction of an SVM intrusion detection system consists of three phases:

* Preprocessing: using automated parsers to process the randomly selected raw TCP/IP dump data in to machine-readable form.
* Training: in this process SVM is trained on different types of attacks and normal data. The data have 41 input features and fall into two classes: normal (+1) and attack (-1).
* Testing: measure the performance on testing data.

A. 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 view the classification problem as a quadratic optimization problem. They combine generalization control with a technique to avoid the "curse of dimensionality" by placing an upper bound on the margin between the different classes, making it a practical tool for large and dynamic data sets. 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 feature space [12].

The SVMs are based on the idea of structural risk minimization, which minimizes the generalization error, i.e. true error on unseen examples. 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 overfitting. 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, such as a linear, polynomial, or sigmoid curve, to the SVMs during the training process, which selects support vectors along the surface of this function. This capability allows classifying a broader range of problems. The primary advantage of SVMs is binary classification and regression that they provide to a classifier with a minimal VC-dimension [12], which implies low expected probability of generalization errors. In our case all intrusions are classified as +1, and normal data are classified as –1. All the SVMs experiments described below use the freeware package SVM light [13].

There are two main reasons that we experiment with SVMs for intrusion detection. The first is speed: as real-time performance is of primary importance to intrusion detection systems, any classifier that can potentially outrun neural networks 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 [14], so they can potentially learn a larger set of patterns and be able to 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 [14].

B. The development of SVM IDS

The data is first partitioned into two classes: normal and attack, where attack represents a collection of 22 different attacks belonging to the four classes described in section 3.1. The objective is to separate normal (1) and intrusive (–1) patterns.

1) Training: The SVMs are trained with normal and intrusive data. Our processed data consists of 14292 data points: 7312 for training, 6980 for testing. Each point is located in the n-dimensional space, with each dimension corresponding to a feature of the data point. We used a training set of 7312 data points with 41 features [16]. Data points contain actual attacks and normal usage patterns. Data points are used for training using the RBF (radial bias function) kernel option; an important point of the kernel function is that it defines the feature space in which the training set examples will be classified [13].

During the training process the default regularization parameter is set to c = 1000, with optimization done for 2733 iterations. During training only 6 data points from the 7312 training set are misclassified. A difference of 0.00072 was achieved with the CPU run time of 17.77 seconds. The number of support vectors used in the training process were 204, including 29 at the upper bound. Linear loss during the process was 17.78295. The normalization of the weight vector (w) during the training process is 126.10847; normalization of the longest example vector (x) is 1.0000. The number of kernel evaluations is 3148450. The estimated VC-dimension [12] of the classifier is less than or equal to 31807.69124.

2) Testing: We apply SVMs to a set of intrusion data as described in Section 3 above. In our case we use the SVMs to differentiate intrusions and normal activities. The testing set, consisting of 6980 data points with 41 features, received 99.50% accuracy, with a total runtime of 1.63 sec. The following graph shows the results.

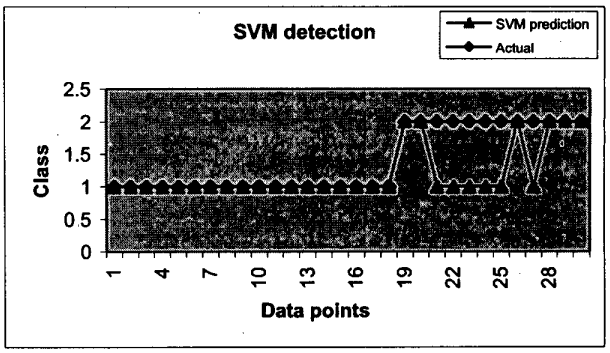

Fig. 1. SVMs results on KDD intrusion detection (outputs 1 denote 1 or normal; outputs 2 denote –1 or attack)

V. The Neural Network Intrusion Detection System

The neural network intrusion detection system consists of three phases:

* Using automated parsers to process the raw TCP/IP dump data in to machine-readable form.
* Training: neural network is trained on different types of attacks and normal data. The input has 41 features and the output assumes one of two values: intrusion (22 different attack types), and normal data.
* Testing: performed on the test set containing 6980 data.

A. Experiments Using Neural Networks

Multi-layer, feed-forward networks are used. The scaled conjugate gradient descent algorithm, available from the MATLAB package, is used for training.

Our data consists of the same set of 14292 data points. The set of 7312 training data is divided in to two classes: normal and attack, where the attack is a collection of 22 different types of instances that belong to the four classes described in section 3, and the other is the normal data.

In the study we use three different feed-forward neural networks with the following architectures:

Network A: 4-layer, 41-20-20-20-1.
Network B: 3-layer, 41-40-40-1.
Network C: 3-layer, 41-25-20-1.

We use an initial training set of 7312 normalized input-output pairs consisting attack patterns, and normal user patterns.

1) Training Neural Networks: The training of the neural networks was conducted using feed forward back propagation algorithm using scaled conjugate gradient decent or SCG for learning. The network was set to train until the desired mean square error of 0.001 was met. During the training process the goal was met at 394 epochs with a performance of 0.0009962988, Fig 2 shows the training process. The purpose of having multiple networks is to find a suitable architecture that can detect at a faster speed with low error rate, minimizing false positives and false negatives. Out of all the networks architectures used, network B performs the best detection with 99.25% accuracy.

Figure 2 below demonstrates extremely good results of the training of network B that converges in 394 epochs, while other methods we tried (Gradient Descent with Adaptive Linear Back Propagation and Gradient Descent with Momentum and Adaptive Linear Back Propagation) took longer.

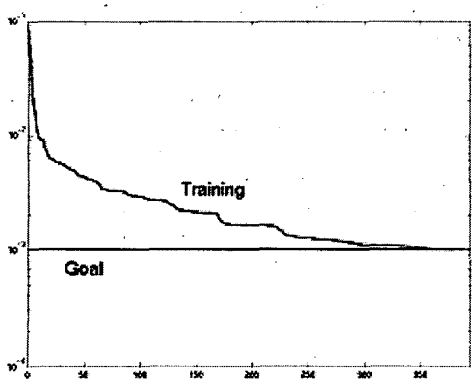

Fig. 2. Neural network training on KDD intrusion detection data-subset

2) Testing the Neural Network: The testing set, as before, consisting of 6980 data points with 41 features. We have three different feed-forward, multi-layer neural network architectures. The following figure shows the results of three different architectures: Network A performed with an accuracy of 99.05%; network B achieved an accuracy of 99.25%; network C performed with an accuracy of 99%.

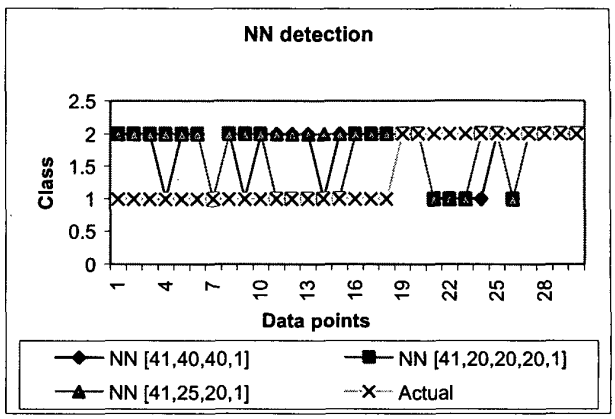

Fig. 3. Neural network testing on KDD intrusion detection

VI. Comparison of SVMs and Neural Networks

Figure 4 below shows a comparison of the results of (the best performing) neural network and support vector machines on the KDD data subset selected for testing. Due to the large size of the testing set, only thirty data points are shown here. As can be seen, the SVM IDS has a slightly higher rate of making the correct detection.

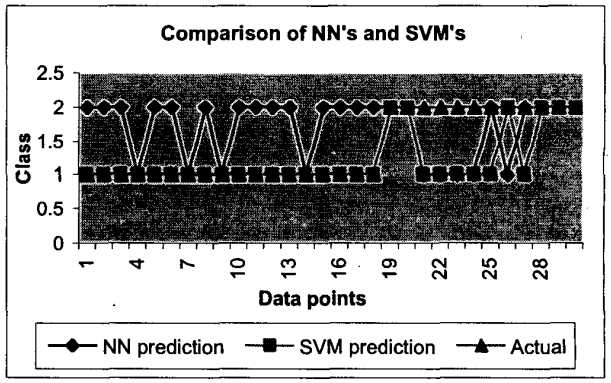

Fig. 4. Neural network and SVMs testing on two classes attack/normal data

VII. Conclusion

We have constructed intrusion detection systems using neural networks and support vector machines, and tested their performance on a set of benchmark DARPA data. It is observed that both the neural networks and SVMs deliver highly accurate results (greater than 99% accuracy on testing set) and show compatible level of performance. The training time for SVMs is significantly shorter (17.77 sec vs. 18 min), an advantage that becomes rather important in situations where retraining needs to be done quickly (e.g., when new attack patterns are discovered). The running time of SVMs is also notably shorter. On the other hand, SVMs can only make

binary classifications, which is a severe disadvantage where the intrusion detection system requires multiple-class identifications (e.g., all 22 different types of attacks need to be differentiated).

Statistical learning techniques are being used more extensively in recent intrusion detection systems, owing to their adaptability and their generalization capability regarding new attack signatures that would need to be 'learned' quickly? once discovered? by an IDS. Whether to use SVMs or neural networks in implementing an intrusion detector depends on the particular type of intrusion (anomaly or misuse) that is under watch, as well as other security policy requirements. SVMs have great potential to be used in place of neural networks due to its scalability (large data sets and large number of features in patterns can easily overwhelm neural networks) and faster training and running time. On the other hand, neural networks have already proven to be useful in many IDSs, and are especially suited for multi-category classifications.

VIII. Acknowledgements

Partial support for this research received from ICASA (Institute for Complex Additive Systems Analysis, a division of New Mexico Tech) is gratefully acknowledged. The second author also acknowledges her partial support received from Sandia National Laboratories under the Rio Grande Educational Initiative. We would also like to acknowledge many insightful conversations with Dr. Jean-Louis Lassez, David Duggan, and Bob Hutchinson that contributed greatly to our work.

IX. References

[1] Ryan J, Lin M-J, Miikkulainen R (1998) Intrusion Detection with Neural Networks. Advances in Neural Information Processing Systems 10, Cambridge, MA: MIT Press.

[2] Kumar S, Spafford EH (1994) An Application of Pattern Matching in Intrusion Detection. Technical Report CSD-TR-94-013. Purdue University.

[3] Luo J, Bridges SM (2000) Mining Fuzzy Association Rules and Fuzzy Frequency Episodes for Intrusion Detection. International Journal of Intelligent Systems, John Wiley & Sons, pp 15:687-703.

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

[5] Sung AH (1998) Ranking Importance Of Input Parameters Of Neural Networks. Expert Systems with Applications, pp 15:405-411.

[6] Cramer M, et. al. (1995) New Methods of Intrusion Detection using Control-Loop Measurement. Proceedings of the Technology in Information Security Conference (TISC) '95. pp 1-10.

[7] Debar H, Becke M, Siboni D (1992) A Neural Network Component for an Intrusion Detection System. Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy.

[8] Debar H, Dorizzi B (1992) An Application of a Recurrent Network to an Intrusion Detection System. Proceedings of the International Joint Conference on Neural Networks. pp 78-483.

[9] Denning D (FEB 1987) An Intrusion-Detection Model. IEEE Transactions on Software Engineering, Vol. SE-13, No 2..

[10] Ghosh AK. (1999). Learning Program Behavior Profiles for Intrusion Detection. USENIX.

[11] Cannady J. (1998) Artificial Neural Networks for Misuse Detection. National Information Systems Security Conference.

[12] Vladimir VN (1995) The Nature of Statistical Learning Theory. Springer, Berlin Heidelberg New York.

[13] 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).

[14] Joachims T (1998) Making Large-Scale SVM Learning Practical. LS8-Report, University of Dortmund, LS VIII-Report.

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

[16] http://kdd.ics.uci.edu/databases/kddcup99/task.html.