Adaptive Neuro-Fuzzy Intrusion Detection Systems

Sampada Chavan, Khusbu Shah, Neha Dave and Sanghamitra Mukherjee
Institute of Technology for Women, SNDT University, India
Ajith Abraham
Department of Computer Science, Oklahoma State University, USA
Sugata Sanyal
School of Tech. and Computer Science, Tata Institute of Fundamental Research, India

Abstract

The Intrusion Detection System architecture commonly used in commercial and research systems have a number of problems that limit their configurability, scalability or efficiency. In this paper, two machine-learning paradigms, Artificial Neural Networks and Fuzzy Inference System, are used to design an Intrusion Detection System. SNORT is used to perform real time traffic analysis and packet logging on IP network during the training phase of the system. Then a signature pattern database is constructed using protocol analysis and Neuro-Fuzzy learning method. Using 1998 DARPA Intrusion Detection Evaluation Data and TCP dump raw data, the experiments are deployed and discussed.

1. Introduction

Knowledge is the first line of defense against any security threat. Forewarned is to be forearmed. Advances in networking and the continued spread of the Internet are adding to the ranks of malicious hackers as well as facilitating information flow [13]. But, security and intrusion detection procedures are also growing in sophistication [6][15]. An Intrusion Detection System (IDS) is a computer program that attempts to perform ID by either misuse or anomaly detection, or a combination of techniques.

2. Proposed System Architecture

We divide the work-system to contain the following distinct parts as depicted in Figure 1:
LAN: A connection of nodes using an Ethernet topology, a trusted network.

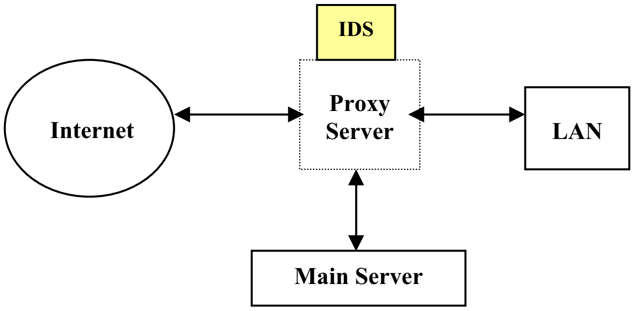

Figure 1. Architecture of the proposed framework

Host/ server: For any internal network
Internet: The gateway to the external world.
Proxy Server: In an enterprise that uses the Internet, a proxy server is a server that acts as an intermediary between a workstation user and the Internet so that the enterprise can ensure security, administrative control, and caching service.
IDS: Compared to the IDS models available in the literature [11][12], the proposed IDS could learn new types of attacks continuously without erasing the previously trained knowledge. Based on the available new data, IDS could update the knowledge base using simple *if-then* fuzzy rules. For the smooth running of the system, the following software components should be present.

2.1. Proxy server

The proxy server is a kind of a service that examines what application or service a packet is meant for and if that particular service is available, only then is the packet allowed to pass through. Thus there is no direct connection between the untrusted and trusted system [14].

First, the proxy server acts as an intermediary, helping users on a private network get information from the

0-7695-2108-8/04 $ 20.00 © 2004 IEEE

Internet, while ensuring that network security is maintained. Second, a proxy server may store frequently requested information in a local disk cache, rapidly delivering it to multiple users without having to go back to the Internet to get it [14].

Proxy servers perform network address translation, mapping all of a network's internal IP addresses to a single safe IP address.

2.2. Intrusion Detection System

The network based intrusion detection system consists of 3 subsystems. The input subsystem include, snort IDS as packet sniffer and signature database, the processing subsystem makes use of misuse and anomaly based detection techniques in combination with neural networks to make it adaptive. The output subsystem uses various reporting mechanisms.

2.2.1. SNORT

By using snort we can access the data that is essential as inputs to the algorithm for its training phase. Snort is used only during the training phase of the algorithm. SNORT is a libpcap-based sniffer and logger [3]. It is a cross-platform, lightweight intrusion detection tool that can be deployed to monitor small TCP/IP networks and detect a wide variety of suspicious network traffic as well as outright attacks. It can provide administrators with enough data to make informed decisions on the proper course of action in the face of suspicious activity. The detection engine is programmed using a simple language that describes per packet tests and actions. The major feature that SNORT has is packet payload inspection. SNORT decodes the application layer of a packet and can be given rules to collect traffic that has specific data contained within its application layer.

2.2.2. Signature Database and Protocol Analysis

For achieving the purpose we also consider another reference base that we build on our own. This base would consist of signature patterns, which try to incorporate cures to the vulnerabilities present in snort. A network IDS signature is a pattern that we want to look for in traffic. We review some examples and some of the methods that can be used to identify signatures.

- Connection attempt from a reserved IP address is identified by checking the source address field in an IP header and packets with an illegal TCP flag combination. This can be found by comparing the flags set in a TCP header against known good or bad flag combinations (email containing a particular virus).
- The IDS can compare the subject of each email to the subject associated with the virus-laden email, or it can look for an attachment with a particular name.
- Denial of service attack on a POP3 server caused by issuing the same command thousands of times. One signature for this attack would be to keep track of how many times the command is issued and to alert when that number exceeds a certain threshold.
- File access attack on an FTP server by issuing file and directory commands to it without first logging in. A state-tracking signature could be developed which would monitor FTP traffic for a successful login and would alert if certain commands were issued before the user had authenticated properly [4].

Because we've identified four potential signature elements, we have many different options for developing a header-based signature, because a signature could include any one or more of these characteristics. However, a signature based on all four suspicious characteristics may be too specific. Although it would provide much more precise information about the source of the activity, it would also be far less efficient than a signature that only checks one header value. Signature development is always a tradeoff between efficiency and accuracy. In many cases, simpler signatures are more prone to false positives than more complex signatures, because simpler signatures are much more general. But more complex signatures may be more prone to false negatives than simpler signatures [4].

An intrusion detection signature set is much more valuable if it can detect not only known attacks, but also future and unknown attacks. Even though the characteristics of the traffic are changing, we can still identify it as anomalous through the use of more general signatures. For this we focus on our strategy of using protocol analysis to look for more general signatures and beyond that we look for developing a self-learning IDS using AI paradigms.

The term "*protocol analysis*" means that the IDS sensor understands how various protocols work and closely analyze the traffic of those protocols to look for suspicious or abnormal activity [2]. Protocol analysis techniques observe all traffic involving a particular protocol and validate it, alerting when the traffic does not meet expectations. Several header values can be used to create network IDS signatures. Some of the most commonly used header-related signature elements are [4]:

- IP addresses (particularly reserved, non-routable, and broadcast addresses)
- Port numbers that should not be in use (well-known ports for particular protocols and Trojans)
- Unusual packet fragmentation
- Particular TCP flag combinations
- ICMP types/codes that should not normally be seen

By focusing on anomalies within the traffic, rather than simply looking for the signatures of particular exploits, protocol analysis-based signatures are much more difficult for attackers to evade through changes to exploits' code or IDS obfuscation techniques.

2.2.3. Machine Learning Paradigms

Various nonlinear systems have been proposed for retrieving desired or stored patterns. The results can be either computed in one epoch or updated iteratively based on the retrieving dynamics equations. Our approach is to develop adaptive machine learning algorithms to develop IDS. A salient feature of Artificial Neural Networks (ANN) is their learning ability. They learn by adaptively updating the synaptic weights that characterize the strength of the connections. The weights are updated according to the information extracted from new training patterns.

A Fuzzy Inference System (FIS) can utilize human expertise by storing its essential components in rule base and database, and perform fuzzy reasoning to infer the overall output value. The derivation of *if-then* rules and corresponding membership functions depends heavily on the *a priori* knowledge about the system under consideration. However there is no systematic way to transform experiences of knowledge of human experts to the knowledge base of a FIS. There is also a need for adaptability or some learning algorithms to produce outputs within the required error rate.

To a large extent, the drawbacks pertaining to these two approaches seem complementary. Therefore it is natural to consider building an integrated system combining the concepts of FIS and ANN modeling [1]. Evolving Fuzzy Neural Network (EFuNN) implements a Mamdani type FIS [9] and all nodes are created during learning [7]. The nodes representing membership functions (MF) can be modified during learning. Each input variable is represented here by a group of spatially arranged neurons to represent a fuzzy quantization of this variable. Different membership functions can be attached to these neurons (triangular, Gaussian, etc.). New neurons can evolve in this layer if, for a given input vector, the corresponding variable value does not belong to any of the existing MF to a degree greater than a membership threshold. A new fuzzy input neuron, or an input neuron, can be created during the adaptation phase of an EFuNN. Technical details of the learning algorithm are given in [7].

3. Proposed System Implementation

3.1. Phase 1: Training the algorithms

For any machine learning based algorithm we require a good training dataset to get the optimal solution. We train by giving a huge set of inputs (which may or may not be attacks) and the corresponding outputs raised in each case. We make use of snort to get access to the data, which is essential to be fed as inputs during the training phase. With the huge database of attacks that the ANN and FIS accumulate during the training phase we expect it to be capable of identifying attacks based on its developed knowledge base in future. But our aim is to provide a system that nears the characteristics of an ideal IDS i.e. minimising the number of false alarms.

3.2. Phase 2: The Execution

For achieving the above purpose we consider another reference base that we build on our own. This base consists of signature patterns that would help encounter vulnerabilities present in snort. Our inputs will be the parameters retrieved from the tcp dump. We would look for match patterns from what the machine learning algorithms have learnt and acquired in the database along with the signature database. In this case we would be able to specifically determine whether an attempt is an attack or a normal packet. In the process we are reducing the number of false positives and false negatives. Also we can add signatures corresponding to the new vulnerabilities and enhance the database.

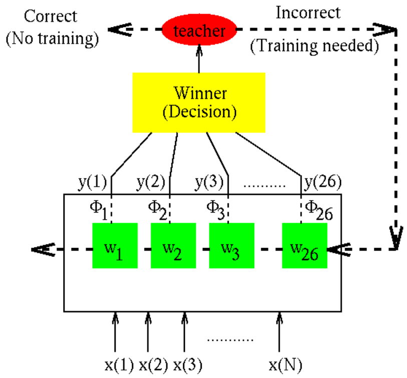

Figure 2. Multi-class attack learning framework

3.3. Neural Network Based Learning

One subnet is designated for each class of attack (Figure 2). The linear discriminant functions for the subnets are denoted as $\Phi\ (x,w_i)$, for $i = 1, ..., L$. The discriminant function provides the score for each subnet (or each class). A procedure is then used to select the subnet (or class) with the winning score. The output is usually a symbol labeling the winner of the subnets.

The following mutual training scheme is used [3][5]. If the desired outputs match the network produced output, then the network will be left alone until a future training pattern is presented. If the net mismatch, then the weights will be updated by reinforcement and anti-reinforcement learning rules. Suppose that $S = \{ x^{(1)} , \ldots , x^{(M)} \}$ is a set of given training patterns, with each element $z^{(m)}$ Є R^N belonging to one of the L classes $\{ \Omega_i , I = 1 , \ldots , L \}$; and that the discriminant functions are $\Phi\ (x,w_i) = w^T_i\ z$

for i = 1, ..., L. Suppose that the m^{th} pattern $x^{(m)}$ presented is known to belong to class Ω_i ; and that the winning class for the pattern is denoted by an integer j, that is, for all $l \neq j$,
$w^T_j > w^T_l z$

1.When j=i, then the pattern $z^{(m)}$ is already correctly classified, so no update will be needed.

2.When $j \neq i$,that is, $z^{(m)}$ is still misclassified, then the following update will be performed:

Reinforcement Learning: $w^{(m+1)}_i = w^{(m)}_i + \eta z^{(m)}$
Anti-reinforcement Learning: $w^{(m+1)}_j = w^{(m)}_j - \eta z^{(m)}$
The other weights remain unchanged:
$w^{(m+1)}_l = w^{(m)}_l$ for all $l \neq i$ and $l \neq j$.

4 Experiment Setup and Results

To simulate the presented ideas, we used the 1998 DARPA Intrusion Detection Evaluation program data provided by MIT Lincoln Labs [10]. The TCP dump raw data was processed into connection records, which are about five million connection records. The data set contains 24 attack types. The attacks fall into four main categories as follows.

1. **Denial of Service (DOS):** Attacker makes some computing or memory resources too busy or too full to handle legitimate requests, or denies legitimate users access to a machine.
2. **Remote to User (R2L):** Attacker who does not have an account on a remote machine sends packets to that machine over a network and exploits some vulnerability to gain local access as a user of that machine.
3. **User to Root (U2R):** 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.
4. **Probing:** Attacker scans a network of computers to gather information or find known vulnerabilities. An attacker with a map of machines and services that available on a network can use this information to look for exploits.

The original data contains 744 MB data with 4,940,000 records. The data set has 41 attributes for each connection record plus one class label. Some features are derived features, which are useful in distinguishing normal connection from attacks. Some features examine only the connections in the past two seconds that have the same destination host as the current connection, and calculate statistics related to protocol behavior, service, etc. These are called same host features. Same host and same service features are together called time-based traffic features of the connection level records. Our initial research was to reduce the number of variables. Using all the 41 variables could result in a big IDS model, which could be an overhead for online detection. We generated a decision tree to determine the variable importance. Variable importance for a particular predictor (attack) is the sum across all nodes in the tree of the improvement scores that the predictor has when it acts as a primary or surrogate (but not competitor) splitter. Example: for node *n* if the predictor appears as the primary splitter then it has a contribution X towards importance. If instead the predictor appears as the n^{th} surrogate instead of primary predictor the contribution becomes $p^n \times X$, where p is the surrogate improvement weight (could beset anywhere between 0 and 1). The main purpose of IDS model is to classify the data set into one of the four attack types or normal. The data set for our experiments contained 11982 records, which are randomly generated from the master data set [8]. This data set has five different classes, random generation of data include the number of data from each class proportional to its size, except that the smallest class is completely included. This data set is again divided into training data with 5092 records and testing with 6890 records. All IDS models are trained and tested with the same set of data. The experiment setup consists of two stages: Network training and performance evaluation. All the training data were scaled to (0-1). The decision tree approach helped us to reduce the number of variables to 13, 14, 15, 17, and 16 respectively for *Normal, DOS, U2L, U2R* and *Probes*.

- **EFuNN training**

We used 4 (MF) membership functions for all the input variables and the following evolving parameters for detection all the classes of attacks: sensitivity threshold *Sthr*=0.95 and error threshold *Errthr*=0.05. During training, we developed 89, 115, 123, 134 and 129 rule nodes for *Normal, DOS, U2L, U2R* and *Probes*.

- **ANN training**

We used a network with 80 hidden neurons and the number input neurons corresponding to the input variables and 1 output neuron. Initial weights, learning rate and momentum used were 0.3, 0.1 and 0.1, respectively. The training was terminated after 4500 epochs.

Table 1 illustrates the comparative performance (classification accuracy of the different attack types) between EFuNN and ANN on the test dataset. While EFuNN took few seconds to train the IDS models, ANN took few minutes to converge. Except U2R, the developed fuzzy inference system could detect with high accuracy. The performance was degraded when we used all the 41 variables, which also illustrates the importance of input variable selection. Due to space restrictions, the complete results are not provided in this paper.

An important advantage of the developed FIS based IDS is its easy interpretability using simple *if-then* rules.

These rules could be learned automatically from the data, which makes EFuNN an ideal candidate for adaptive learning. Using Protocol analysis, we can add signatures corresponding to the new vulnerabilities and enhance the database and keep on improving the IDS performance. It will not only search for match patterns from what it has learnt and acquired in the database but also with the signature database which would in turn help in coming up with an even better optimal result. We are thus reducing the number of false positives and false negatives. There is no direct connection between the untrusted system and the trusted system as the packets are scrutinized by the IDS at the proxy server.

Type of attack	Classification Accuracy %	
	EFuNN	ANN
Normal	99.56	99.57
Probe	99.88	94.62
DOS	98.99	98.97
U2R	65.00	59.00
R2L	97.26	97.02

Table 1. Performance comparison using reduced number of input variables

EFuNN uses a hybrid learning technique (a mixture of unsupervised and supervised learning) to fine-tune the parameters of the FIS. As EFuNN adopts a single pass training (1 epoch) it is more adaptable and easy for further on-line training, which might be highly useful for online detection and updating the knowledge base. Another important feature of EFuNN is that the user has the flexibility to construct the network (by selecting the parameters).

5. Conclusions

We have demonstrated the use of two machine-learning paradigms for designing IDS. EFuNN performed well compared to neural networks. Experiment results also reveal the importance of input variable reduction. By having less than 40% of the original number of input variables, we are able to improve the performance and development time.

The future of IDS lies in data correlation. The IDS of tomorrow will produce results by examining input from several different sources. The way to solve this challenge lies in statistical analysis and predictive artificial intelligence performed on strange data sets. Intrusion Detection Systems face several daunting, but exciting challenges in the future and are sure to remain one of our best weapons in the arena of network security.

References

[1] Abraham A., Neuro-Fuzzy Systems: State-of-the-Art Modelling Techniques, Lecture Notes in Computer Science. Volume. 2084, Springer-Verlag Germany, Jose Mira and Alberto Prieto (Eds.), pp. 269-276, 2001.

[2] Mukherjee B., Heberlein T.L. and Levitt K.N., Network intrusion detection. IEEE Network, 8(3): 26, 1994.

[3] Lau C., Neural Networks, Theoretical Foundations and Analysis, IEEE Press, 1991

[4] Mark C. et al, Intrusion Signatures and Analysis, SANS Giac, 2002 Reprint

[5] Fausett L., Fundamentals of Neural Networks, Prentice Hall, 1994.

[6] Karen F.K., Network Intrusion Detection Signatures, securityfocus.com, December 19, 2001

[7] Kasabov N., Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation, in Yamakawa T and Matsumoto G (Eds), Methodologies for the Conception, Design and Application of Soft Computing, World Scientific, pp. 271-274, 1998.

[8] KDD cup 99 Intrusion detection data set. <http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz>

[9] Mamdani E.H. and Assilian S., An experiment in Linguistic Synthesis with a Fuzzy Logic Controller, International Journal of Man-Machine Studies, Vol. 7, No.1, pp. 1-13, 1975.

[10] MIT Lincoln Laboratory. <http://www.ll.mit.edu/IST/ideval/>

[11] Mukkamala S., Sung A.H. and Abraham A., Distributed Multi-Intelligent Agent Framework for Detection of Stealthy Probes, Design and Application of Hybrid Intelligent Systems, Abraham A., Köppen M. and Franke K. (Eds.), IOS Press, Amsterdam, The Netherlands, pp. 116-125, 2003.

[12] Mukkamala S., Sung A.H. and Abraham A., Intrusion Detection Using Ensemble of Soft Computing Paradigms, Intelligent Systems Design and Applications, Abraham A., Köppen M. and Franke K. (Eds.), Springer Verlag, Germany, pp. 239-248, 2003.

[13] IEEE Journal on Selected Areas in Communications May 1989. Special issue on Secure Communications.

[14] Jonathan A., Proxy servers, Network Magazine, 04/01/1999 < http://www.networkmagazine.com/>

[15] Brian C. et al, Snort 2.0 Intrusion Detection, Paperback - February 2003