RESEARCH ARTICLE OPEN ACCESS

A Framework for Intrusion Detection System

Bonealli Pedda Uppalaiah
Department of Computer Science & Engineering
HITS COE, and Hyderabad
India

ABSTRACT

Intrusion Detection systems are now an important component in the overall network and data security system. With the rapid advancement in the network technologies including greater bandwidths and ease of connectivity of wireless and mobile devices, the focus of intrusion detection has shifted from simple autograph harmonizing approaches to detecting attacks based on analyzing background information which may be unambiguous to individual networks and applications. As a result, anomaly and hybrid intrusion detection approaches have gained importance. However, present anomaly and hybrid detection approaches suffer from some major setbacks; limited attack detection coverage, large number of fake alarms. In this paper, we discuss layered approach which is effective in detecting a wide selection of attacks and which result in very few fake alarms. Additionally, using proposed approach, attacks can not only be accurately detected but can also be identified which helps to initiate effective intrusion response mechanisms in real-time systems.

Keywords :—Attacks, fake alarms, Intrusion detection, Layered Approach, network security.

I. INTRODUCTION

As network-based computer systems play increasingly vital roles in recent society, they have grow to be the targets of our enemies and criminals. Therefore, we need to find the best ways achievable to protect our systems. The security of a computer system is compromised when an intrusion takes place. An intrusion can be defined as "any set of proceedings that attempt to compromise the integrity, privacy or accessibility of a resource". In addition to intrusion prevention techniques, such as user authentication (e.g. using passwords or biometrics), avoiding programming errors, and information protection (e.g., encryption), intrusion detection is often used as another wall to protect computer systems. Intrusion detection as defined by the SysAdmin, Audit, Networking, and Security (SANS) Institute is the art of detecting badly chosen, incorrect, or anomalous doings [1]. The aim of an intrusion detection system is to provide data security and make sure continuity of services provided by a network. Given the diverse type of attacks (Denial of Service, Probing, Remote to Local, User to Root and others), it is a test for any intrusion detection system to detect a wide variety of attacks with very few fake alarms in real-time atmosphere. Ideally, the system must detect all intrusions with no fake alarms. The challenge is, thus, to construct a system which has broad attack detection coverage and at the same time which results in very few fake alarms. However, this is in no way a solution for securing today's highly networked computing environment and, hence, the need to develop better intrusion detection systems.

II. LITERATURE REVIEW

The field of intrusion detection and network security has been around since late 1980s. Since then, a number of methods and frameworks have been proposed and many systems have been built to detect intrusions. Various techniques such as association rules, clustering, naive Bayes classifier, artificial neural networks, and others have been applied to detect intrusions. In this section, we briefly discuss these techniques and frameworks.

Data mining approaches for intrusion detection include association rules and frequent episodes, which are based on construction classifiers by discovering related patterns of program and user behavior [3] AND [4]. Association rules and frequent episodes are used to learn the record patterns that describe user behavior. These methods can deal with symbolic data, and the features can be defined in the form of packet and connection details. Data clustering methods such as the k-means and the fuzzy c- means have also been applied extensively for intrusion detection [5]. One of the main drawbacks of the clustering technique is that it is based on calculating numeric distance between the clarification, and hence, the observations must be numeric. clarification with symbolic features cannot be easily used for clustering, resulting in incorrectness.

Naive Bayes classifiers have also been used for intrusion detection [6]. However, they make authoritarian independence assumption between the features in an examination resulting in lower attack detection correctness when the features are correlated, which is often the case for intrusion detection. Bayesian network can also be used for intrusion detection [7]. However, they tend to be attack specific and construct a decision network based on special characteristics of individual attacks. Thus, the size of a Bayesian network increases speedily as the number of features and the type of attacks modelled by a Bayesian network increases.

Decision trees have also been used for intrusion detection [6]. The decision trees select the best features for each decision node during the construction of the tree based on some well-defined criteria. One such criterion is to use the information

gain ratio, which is used in C4.5. Decision trees generally have very high speed of operation and high attack detection accuracy.

Debar et al. [8] and Zhang et al. [9] discuss the use of artificial neural networks for network intrusion detection. Though the neural networks can work effectively with noisy data, they require large amount of data for training and it is often tough to select the best achievable architecture for a neural network.

We balance the Layered Approach with the decision tress, naive Bayes classification methods. Our system is based upon serial layering of multiple hybrid detectors. The results from individual classifiers at a layer are not combined at any later period in the Layered Approach, and therefore, an attack can be blocked at the layer where it is detected. There is no communication transparency among the layers and the central decision-maker. In addition, since the layers are self-determining they can be trained separately and deployed at critical locations in a network depending upon the specific requirements of a network.

TABLE I
Existing Intrusion Detection Methods

Detection Method	Features	Drawbacks
Data clustering	It is based on calculating numeric distance between the observations, and hence, the observations must be numeric.	Observations with symbolic features cannot be easily used for clustering, resulting in inaccuracy.
		The clustering methods consider the features independently and are unable to capture the relationship between different features of a single record, which further degrades attack detection accuracy [5].
Data mining	Data mining approaches for intrusion detection include association rules and frequent episodes	Data mining requires the number of records to be large and sparsely populated; otherwise they tend to produce a large number of rules that increase the complexity of of the system [3] and [4].
	These methods can deal with symbolic data, and the features can be defined in the form of packet and connection details.	
Naive Bayes classifiers	Bayesian network can also be used for intrusion detection.	It makes strict independence assumption between the features in an observation resulting in lower attack detection accuracy.
		The size of a Bayesian network increases rapidly as the number of features and the type of attacks modeled by a Bayesian network increases [7].
Artificial neural networks	The neural networks can work effectively with noisy data	Though the neural networks can work effectively with noisy data, they require large amount of data for training and it is often hard to select the best possible architecture for a neural network [8].

III. SOFTWARE ARCHITECTURE AND WORK FLOW

In this Section, a layered framework is planned for construction anomaly and hybrid network intrusion detection systems which can control powerfully in high speed networks and can correctly detect a variety of attacks. Our proposed framework is very general and can be easily adapted by adding domain specific knowledge as per the specific requirements of the network in concern, thereby, giving flexibility in execution. Figure 1 represents framework for building Layer based Intrusion Detection Systems (LIDS). The figure represents an ‘ n’ layer system where every layer in itself is a small intrusion detection system which is particularly trained to detect only a single type of attack, for example the DoS attack. A number of such sub systems are then deployed in order, one after the other. This serves dual purpose; first, every layer can be skilled with only a small number of features which are important in detecting a particular class of attack. Second, the size of the sub system remains small and hence, it performs efficiently.

A common disadvantage of using a modular approach, similar to our layered framework, is that it increases the communication transparency among the modules (sub systems). However, this can be easily eliminated in our framework by making every layer completely self-determining of every other layer. As a result, some features may be present in more than one layer. Depending upon the security policy of the network, every layer can simply block an attack once it is detected without the need of a central decision maker. A number of such layers fundamentally act as filters, which blocks anomalous connection as soon as they are detected in a particular layer, thereby as long as a quick response to intrusion and simultaneously reducing the analysis at ensuing layers. It is important to note that a different response may be initiated at different layers depending upon the class of attack the layer is trained to detect. The amount of audit data analysed by the system is additional at the first layer and decreases at subsequent layers as more and more attacks are detected and blocked. In the worst case, when no attacks are detected until at the last layer, all the layers have the same load. However, the overall load for the average case is predictable to be much less since attacks are detected and blocked at every subsequent layer. On the contrary, if the layers are set in parallel somewhat than in a order, the load at every sub system is same and is equal to that of the bad case in the in order arrangement. Additionally, the initial layers in the sequential configuration can be replicated to perform load balancing in order to improve performance.

The layered framework is very general and the number of layers in the overall system can be adjusted depending upon the individual requirements of the network in concern. Even though the number of layers and the significance of every layer in framework depend upon the target network, every layer has two significant components:

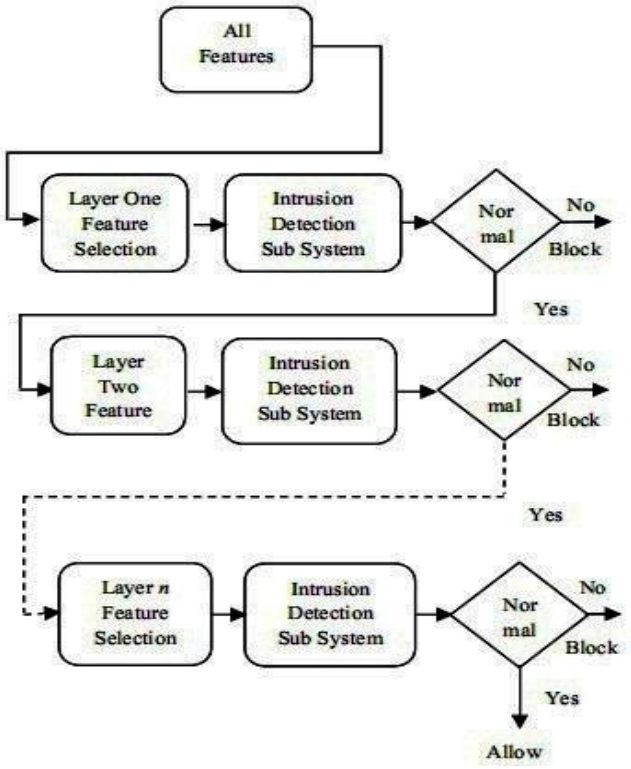

Fig. 1 Layered Framework

A. *Feature Selection Component*

In order to detect intrusions, a large number of features can be monitored. nevertheless, to detect a single attack features such as the 'protocol' and 'type of service' are noteworthy while features such as 'number of root accesses' and 'number of files accessed' are not significant.

B. *Intrusion Detection and Response Sub System*

The second component in every layer is the intrusion detection and answer unit. To detect intrusions, this framework is not restrictive in using a particular anomaly or hybrid detector. A variety of previously well known intrusion detection methods such as the naive Bayes classifier, decision trees, support vector machines and others can be used. A prime advantage of this framework is that newer methods, such as qualified random fields are more effective in detecting attacks can be simply integrated in the framework. Finally, once an attack is detected, the response unit can provide passable intrusion response depending upon the security policy. In order to take advantages of proposed framework, every layer must include both of the above mentioned components.

IV. DETAILED DESIGN

Attacks belonging to different classes are different and, hence for healthier attack detection, it becomes necessary to consider them individually. As a result, in layered system, every layer trained individually to optimally detect a single class of attack. Different features for different layers are chosen based upon the type of attack the layer is trained to detect. In Figure 2, a detailed view of a single layer (Probe layer) is presented which can be used to detect Probe attacks in our system.

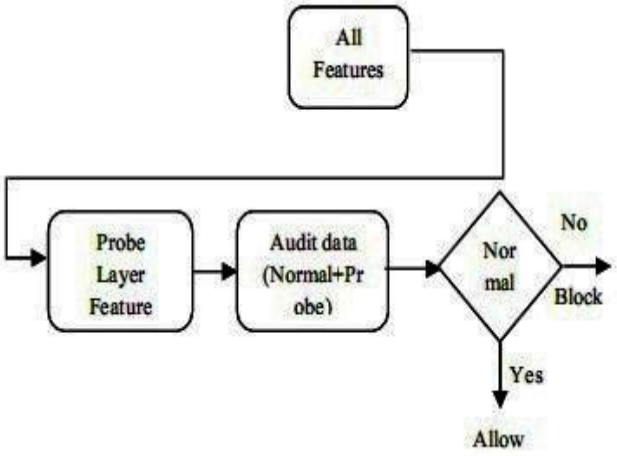

Fig. 2 Representation of Probe Layer with Feature Selection

The Probe layer is optimally trained to detect only the Probe attacks. Hence, only the Probe attacks and the normal instances are used from the audit data to train this layer. Other layers can be trained similarly. Note that, different features are chosen to train different layers in framework. Experimental results clearly recommend that feature selection significantly improves the attack detection ability of the system. Domain knowledge is used to select features for all the four attack classes. An approach for selecting features for every layer is described here:

A. *Probe Layer*

Probe attacks are aimed at acquiring information about the goal network from a source which is often external to the network. Hence, basic connection level features such as the 'duration of connection' and 'source bytes' are significant; while features like 'number of file creations' and 'number of files accessed' are not expected to make available information for detecting Probe attacks

B. *DOS Layer*

DoS attacks are meant to stop the target from providing service(s) to its users by flooding the network with illegal requests. Hence, to detect attacks at the DoS layer; network traffic features such as the 'percentage of connections having same destination host and same service' and packet level features such as the 'source bytes' and 'percentage of packets with errors' are significant. To detect DoS attacks, it may not be important to know whether a user is 'logged in or not' and

hence, such facial appearance are not considered in the DoS layer.

C. R2L Layer

R2L attacks are one of the mainly difficult attacks to detect as they grip both, the network level and the host level features. Hence, to detect R2L attacks, we selected both, the network level features such as the 'duration of connection', 'service requested' and the host level features such as the 'number of unsuccessful login attempts' among others.

D. U2R Layer

U2R attacks involve the semantic details which are very difficult to capture at an early stage at the network level. Such attacks are often content based and end an application. Hence for detecting U2R attacks, we selected features such as 'number of file creations', 'number of shell prompts invoked', while we ignored features such as 'protocol' and 'source bytes'. In the system, every layer is trained individually with the normal instances and with the attack instances belonging to a single attack class. The layers are then approved one after the other in a sequence as shown in Figure 4.3. but, during testing, all the audit patterns (irrespective of their attack class, which is unknown) are accepted into the system starting from the first layer. If the layer detects the instance as an attack, the system labels the instance as a Probe attack and initiates the response mechanism; otherwise it passes the instance to the next layer. Same process is continual at every layer until either an instance is detected as an attack or it reaches the last layer anywhere the instance is labeled as normal if no attack is detected. The algorithm is specified for layered framework.

Algorithm 1 Training
1: Select the number of layers, n, for the complete system.
2: Separately perform features selection for each layer.
3: Train a separate model for each layer using the features selected from Step 2.
4: Plug in the trained models sequentially such that only the connections labeled as normal are passed to the next layer.

Algorithm 2 Testing
1: For each (next) test instance perform Steps 2 through 5.
2: Test the instance and label it either as attack or normal.
3: If the instance is labeled as attack, block it and identify it as an attack represented by the layer name at which it is detected and go to Step 1. Else pass the sequence to the next layer.
4: If the current layer is not the last layer in the system, test the instance and go to Step 3. Else go to Step 5.
5: Test the instance and label it either as normal or as an attack. If the instance is labeled as an attack, block it and identify it as an attack corresponding to the layer name.

V. EXPERIMENTAL WORK

For our experiments, the benchmark KDD '99 intrusion data set [10] can be used. This data set is a version of the original 1998 DARPA intrusion detection assessment program, which is prepared and managed by the MIT Lincoln Laboratory. The data set contains about five million connection records as the training data and about two million connection records as the test data.

The training data is either labeled as normal or as one of the 24 dissimilar kinds of attack. These 24 attacks can be grouped into four classes; Probing, DoS, R2L, and U2R. Similarly, the test data is also labeled as either normal or as one of the attacks belonging to the four attack groups. It is important to note that the test data is not from the same probability allotment as the training data, and it includes specific attack types not present in the training data. This makes the intrusion detection task more sensible.

Weka tool can be used to perform experiments with the decision trees and the naive Bayes classifier. for data formatting and implementing the Layered Approach Java scripts can be used. For all our experiments, we can do hybrid detection, and can use both the normal and the anomalous connections for training the model.

VI. EXPECTED RESULT

As per the results of experiments performed in [2], layered approach improves intrusion detection correctness.

TABLE III
Normal and R2L (All 41 features)

		Precision (%)	Recall (%)	F-Value (%)	Train (Sec)	Test (Sec.)
Naive Bayes	Best	74.10	7.40	13.40	0.38	7.33
	Average	70.03	6.63	12.12		
	Worst	61.30	5.40	10.00		
Decision Trees	Best	98.30	37.10	53.20	0.60	2.75
	Average	84.68	23.39	35.62		
	Worst	63.70	10.40	18.30		

In the experiment about 1,000 normal records are aimlessly selected and all the R2L records from the training data as the training data for detecting R2L attacks. Table 2 gives the results[2]. In Table 3, the testing time of 17.16 seconds represents the time taken to label all the 76,942 test instances. We can watch that the decision trees have a higher F-Value.

TABLE IIIII
Normal and R2L (with Feature Selection)

		Precesion (%)	Recall (%)	F-Value (%)	Train (Sec)	Test (Sec.)
Naive Bayes	Best	74.10	7.40	13.40	0.38	7.33
	Average	70.03	6.63	12.12		
	Worst	61.30	5.40	10.00		
Decision Trees	Best	98.30	37.10	53.20	0.60	2.75
	Average	84.68	23.39	35.62		
	Worst	63.70	10.40	18.30		

Table III gives the results when feature Selection is performed for detecting R2L attacks.

VII. CONCLUSION

In this paper, we have addressed the problem of correctness for construction robust and efficient intrusion

detection systems. Further, we have proposed that feature selection and implementing the Layered Approach can extensively reduce the time required to train and test the model.

REFERENCES

[1] SANS Institute—Intrusion Detection FAQ, http://www.sans.org/ resources/idfaq/, 2010.

[2] Gupta, K.K.; Nath, B.; Kotagiri, R.; , "Layered Approach Using Conditional Random Fields for Intrusion Detection," Dependable and Secure Computing, IEEE Transactions on , vol.7, no.1, pp.35-49, Jan.-March 2010

[3] Wenke Lee; Stolfo, S.J.; Mok, K.W.; , "A data mining framework for building intrusion detection models," Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on , vol., no., pp.120- 132, 1999.

[4] W. Lee, S. Stolfo, and K. Mok, "A Data Mining Framework for Building Intrusion Detection Model," Proc. IEEE Symp. Security and Privacy (SP '99), pp. 120-132, 1999.

[5] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion Detection with Unlabeled Data Using Clustering," Proc. ACM Workshop Data Mining Applied to Security (DMSA), 2001

[6] N.B. Amor, S. Benferhat, and Z. Elouedi, "Naive Bayes vs. Decision Trees in Intrusion Detection Systems," Proc. ACM Symp. Applied Computing (SAC '04), pp. 420-424, 2004.

[7] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, "Bayesian Event Classification for Intrusion Detection," Proc. 19th Ann. Computer Security Applications Conf. (ACSAC '03), pp. 14-23, 2003.

[8] H. Debar, M. Becke, and D. Siboni, "A Neural Network Component for an Intrusion Detection System," Proc. IEEE Symp. Research in Security and Privacy (RSP '92), pp. 240-250, 1992.

[9] Z. Zhang, J. Li, C.N. Manikopoulos, J. Jorgenson, and J. Ucles, "HIDE: A Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification," Proc. IEEE Workshop Information Assurance and Security (IAW '01), pp. 85-90, 2001.

[10] KDD Cup 1999 Intrusion Detection Data. Last accessed: January 30, 2012. http://kdd.ics.uci.edu/databases/ kddcup99/ kddcup99.htm