World Applied Sciences Journal 29 (Computer Sciences, Engineering and Its Applications): 172-176, 2014
ISSN 1818-4952

DOI: 10.5829/idosi.wasj.2014.29.csea.2253

Intrusion Detection and Blocking Using Conditional Random Fields in Layered Approach

K.P. Thooyamani, R. Udayakumar and V. Khanaa

School of Computing Science, Bharath University, Chennai-7, India

Abstract: Intrusion detection faces a number of challenges; an intrusion detection system must reliably detect malicious activities in a network and must perform efficiently to cope with the large amount of network traffic. In this paper, we address the issue of traffic using Conditional Random Fields and Layered Approach. We demonstrate that high attack detection can be achieved by using Conditional Random Fields and high efficiency by implementing the Layered Approach. Experimental results on the benchmark KDD '99 intrusion data set show that our proposed system based on Layered Conditional Random Fields outperforms other well-known methods such as the decision trees and the naive Bayes. Finally, we show that our system is robust and is able to handle traffic without compromising performance.

Key words: Intrusion detection · Layered Approach · Conditional Random Fields · Network security · Decision trees · Naive Bayes

INTRODUCTION

Intrusion detection as defined by the Sys Admin, Audit, Networking and Security (SANS) Institute is the art of detecting inappropriate, inaccurate, or anomalous activity [6]. Today, intrusion detection is one of the high priority and challenging tasks for network administrators and security professionals. More sophisticated security tools mean that the attackers come up with newer and more advanced penetration methods to defeat the installed security systems [4]. Thus, there is a need to safeguard the networks from known vulnerabilities and at the same time take steps to detect new and unseen, but possible, system abuses by developing more reliable and efficient intrusion detection systems. Any intrusion detection system has some inherent requirements. Its prime purpose is to detect as many attacks as possible with minimum number of false alarms, i.e., the system must be accurate in detecting attacks. However, an accurate system that cannot handle large amount of network traffic and is slow in decision making will not fulfill the purpose of an intrusion detection system. We desire a system that detects most of the attacks, gives very few false alarms, copes with large amount of data and is fast enough to make real-time decisions.

Intrusion detection systems are classified as network based, host based, or application based depending on their mode of deployment and data used for analysis [1]. Additionally, intrusion detection systems can also be classified as signature based or anomaly based depending upon the attack detection method. The signature-based systems are trained by extracting specific patterns (or signatures) from previously known attacks while the anomaly-based systems learn from the normal data collected when there is no anomalous activity [2] intrusions is to consider both the normal and the known anomalous patterns for training a system and then performing classification on the test data. Such a system incorporates the advantages of both the signature-based and the anomaly-based systems and is known as the Hybrid System. Hybrid systems can be very efficient, subject to the classification method used and can also be used to label unseen or new instances as they assign one of the known classes to every test instance [3]. This is possible because during training the system learns features from all the classes. The only concern with the hybrid method is the availability of labeled data [4][5]. However, data requirement is also a concern for the signature- and the anomaly-based systems as they require completely anomalous and attack free data, respectively, which are not easy to ensure.

Corresponding Author: R. Udayakumar, School of Computing Science, Bharath University, Chennai-73, India.

Related Work: Various techniques such as association rules, clustering, naive Bayes classifier, support vector machines, genetic algorithms, 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 building classifiers by discovering relevant patterns of program and user behavior. Association rules [8] 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. However, mining of features is limited to entry level of the packet and 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 the system [7]. Data clustering methods such as the k-means and the fuzzy c-means have also been applied extensively for intrusion detection. One of the main drawbacks of the clustering technique is that 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. In addition, 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.

Naive Bayes classifiers have also been used for intrusion detection [9]. However, they make strict independence assumption between the features in an observation resulting in lower attack detection accuracy when the features are correlated, which is often the case for intrusion detection.

Bayesian network can also be used for intrusion detection. However, they tend to be attack specific and build a decision network based on special characteristics of individual attacks. Thus, the size of a Bayesian network increases rapidly as the number of features and the type of attacks modeled by a Bayesian network increases.

To detect anomalous traces of system calls in privileged processes [11], hidden Markov models (HMMs) have been applied in [12]. However, modeling the system calls alone may not always provide accurate classification as in such cases various connection level features are ignored. Further, HMMs are generative systems and fail to model long-range dependencies between the observations. We further discuss this in detail in Section 3.

Decision trees have also been used for intrusion detection [9]. 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.

Conditional Random Field for Intrusion Detection: Conditional models are probabilistic systems that are used to model the conditional distribution over a set of random variables. Such models have been extensively used in the natural language processing tasks. Conditional models offer a better framework as they do not make any unwarranted assumptions on the observations and can be used to model rich overlapping features among the visible observations. Maxent classifiers, maximum entropy Markov models and CRFs are such conditional models. The advantage of CRFs is that they are undirected and are, thus, free from the Label Bias and the Observation Bias. The simplest conditional classifier is the Maxent classifier based upon maximum entropy classification, which estimates the conditional distribution of every class given the observations. The training data is used to constrain this conditional distribution while ensuring maximum entropy and hence maximum uniformity.

In the figure, features such as duration, protocol, service, flag and src_bytes take some possible value for every connection. During training, feature weights are learnt and during testing, features are evaluated for the given observation, which is then labeled accordingly.

As it is evident from the figure, every label is connected to every input feature, which indicates that all the features in an observation help in labeling and thus, a CRF can model dependencies among the features in an observation. Present intrusion detection systems do not consider such relationships among the features in the observations. They either consider only one feature, such as in the case of system call modeling, or assume conditional independence among different features in the observation as in the case of a naive Bayes classifier.

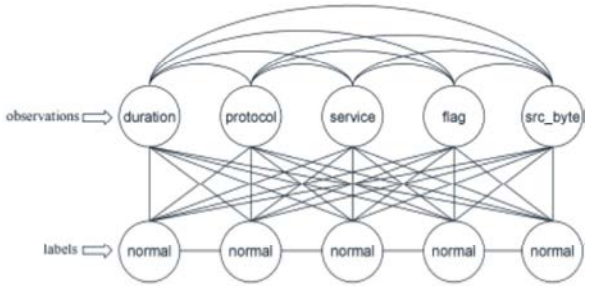

Fig. 1: Graphical representation of a CRF.

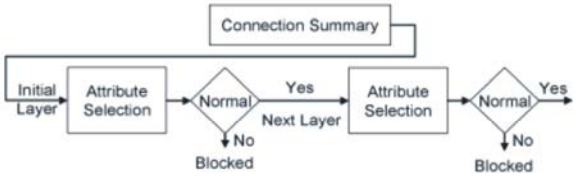

Fig. 2: Layered representation.

Layered Approach for Intrusion Detection: We now describe the Layer-based Intrusion Detection System (LIDS) in detail. The LIDS draws its motivation from what we call as the Airport Security model, where a number of security checks are performed one after the other in a sequence. Similar to this model, the LIDS represents a sequential Layered Approach and is based on ensuring availability, confidentiality and integrity of data and (or) services over a network. Fig. 2 gives a generic representation of the framework.

The goal of using a layered model is to reduce computation and the overall time required to detect anomalous events. The time required to detect an intrusive event is significant and can be reduced by eliminating the communication overhead among different layers. This can be achieved by making the layers autonomous and self-sufficient to block an attack without the need of a central decision-maker. Every layer in the LIDS framework is trained separately and then deployed sequentially. The layers essentially act as filters that block any anomalous connection, thereby eliminating the need of further processing at subsequent layers enabling quick response to intrusion. The effect of such a sequence of layers is that the anomalous events are identified and blocked as soon as they are detected.

Our second goal is to improve the speed of operation of the system. Hence, we implement the LIDS and select a small set of features for every layer rather than using all the 41 features. This results in significant performance improvement during both the training and the testing of the system. In many situations, there is a trade-off between efficiency and accuracy of the system and there can be various avenues to improve system performance. Methods such as naive Bayes assume independence among the observed data. This certainly increases system efficiency, but it may severely affect the accuracy. To balance this trade-off, we use the CRFs that are more accurate, though expensive, but we implement the Layered Approach to improve overall system performance. The performance of our proposed system, Layered CRFs, is comparable to that of the decision trees and the naive Bayes and our system has higher attack detection accuracy.

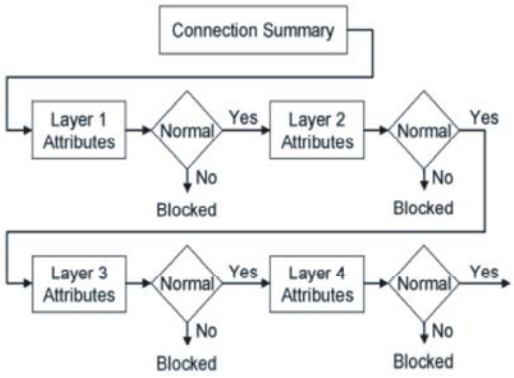

Fig. 3: Real-time representation of the system.

Integrating Layered Approach with Conditional Random Field: As discussed in Sections 3 and 4, respectively, the CRFs can be effective in improving the attack detection accuracy by reducing the number of false alarms, while the Layered Approach can be implemented to improve the overall system efficiency. Hence, a natural choice is to integrate them to build a single system that is accurate in detecting attacks and efficient in operation. Given the data, we first select four layers corresponding to the four attack groups and perform feature selection for each layer, which is described next.

Feature Selection: In this section, we describe our approach for selecting features for every layer and why some features were chosen over others. In our system, every layer is separately trained to detect a single type of attack category. We observe that the attack groups are different in their impact and hence, it becomes necessary to treat them differently. Hence, we select features for each layer based upon the type of attacks that the layer is trained to detect.

Probe Layer: The probe attacks are aimed at acquiring information about the target network from a source that 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 files creations" and "number of files accessed" are not expected to provide information for detecting probes.

DoS Layer: The DoS attacks are meant to force the target to stop the service(s) that is (are) provided by flooding it with illegitimate requests. Hence, for the DoS layer, 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."

R2L Layer: The R2L attacks are one of the most difficult to detect as they involve the network level and the host level features. We therefore selected both the network level features such as the "duration of connection" and "service requested" and the host level features such as the "number of failed login attempts" among others for detecting R2L attacks.

U2R Layer: The U2R attacks involve the semantic details that are very difficult to capture at an early stage. Such attacks are often content based and target an application. Hence, for U2R attacks, we selected features such as "number of file creations" and "number of shell prompts invoked," while we ignored features such as "protocol" and "source bytes.

Algorithm

Training:

Step 1: Select the number of layers, n, for the complete system.

Step 2: Separately perform features selection for each layer.

Step 3: Train a separate model with CRFs for each layer using the features selected from Step 2.

Step 4: Plug in the trained models sequentially such that only the connections labeled as normal are passed to the next layer.

Testing:

Step 5: For each (next) test instance perform Steps 6 through 9.

Step 6: Test the instance and label it either as attack or normal.

Step 7: 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 5. Else pass the sequence to the next layer.

Step 8: If the current layer is not the last layer in the system, test the instance and go to Step 7. Else go to Step 9.

Step 9: 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.

CONCLUSION

Our system can help in identifying an attack once it is detected at a particular layer, which expedites the intrusion response mechanism, thus minimizing the impact of an attack. We showed that our system is robust and performs better than any other compared system. Finally, our system has the advantage that the number of layers can be increased or decreased depending upon the environment in which the system is deployed, giving flexibility to the network administrators. The areas for future research include the use of our method for extracting features that can aid in the development of signatures for signature-based systems. The signature-based systems can be deployed at the periphery of a network to filter out attacks that are frequent and previously known, leaving the detection of new unknown attacks for anomaly and hybrid systems. Sequence analysis methods such as the CRFs when applied to relational data give us the opportunity to employ the Layered Approach, as shown in this paper. This can further be extended to implement pipelining of layers in multicore processors, which is likely to result in very high performance.

REFERENCES

1. Autonomous Agents for Intrusion Detection, http://www.cerias.purdue.edu/researchaafid/, 2010.
2. CRF++: Yet Another CRF Toolkit, http:/crfpp. sourceforge.net/, 2010.
3. KDD Cup, 1999. Intrusion Detection Data, http://kdd. ics.uci.edu/dat abases/kddcup99 kddcup99.html, 2010.
4. Overview of Attack Trends, http://www.cert.org/ archive/pdfattack_trends.pdf 2002.
5. Probabilistic Agent Based Intrusion Detection, http://www.cse.sc.edu/research/islagentIDS.shtml, 2010.
6. SANS Institute-Intrusion Detection FAQ, 2010. http://www.sans.org/resourcesidfaq/.
7. Abraham, T., 2008. IDDM: Intrusion Detection Using DataMining Techniques, http://www.dsto.defence./ gov.au/publications/ 2345DSTO-GD-0286.pdf.

8. Agrawal, R., T. Imielinski and A. Swami, 1993. "Mining Association Rules between Sets of Items in Large Databases," Proc. ACM IGMOD, 22(2): 207-216.
9. Amor, N.B., S. Benferhat and Z. Elouedi, 2004. "Naive Bayes vs. Decision Trees in Intrusion Detection Systems," Proc. ACM Symp. Applied Computing (SAC '04), pp: 420-424.
10. Srinivasan, V. and T. Saravanan, 2013. Sophisticated Technologies in Power Transmission Structure, Middle-East Journal of Scientific Research, ISSN:1990-9233, 15(12): 1809-1812.
11. Udayakumar, R., V. Khanna, T. Saravanan and G. Saritha, 2013. Retinal Image Analysis Using Curvelet Transform and Multistructure Elements Morphology by Reconstruction, Middle-East Journal of Scientific Research, ISSN: 1990-9233, 16(12): 1798-1800.
12. Thooyamani, K.P., V. Khanaa and R. Udayakumar, 2013. An Integrated Agent System for E-mail Coordination using Jade, Indian Journal of Science and Technology, ISSN: 0974-6846, 6(6): 4758-4761.