Proceedings of the 6th WSEAS Int. Conf. on NEURAL NETWORKS, Lisbon, Portugal, June 16-18, 2005 (pp90-94)

Using Self-Organizing Networks for Intrusion Detection

RICHARD A. WASNIOWSKI
Computer Science Department
California State University
Carson, CA 90747, USA

Abstract: - In this paper we propose a framework for using self-organizing networks for agent based intrusion detection system. A specific feature of this model is that the agent uses self organizing algorithm for pattern classification from security logs and self organizing maps for visualization. We have developed a prototype for this framework. This paper discusses also the issues of combining intelligent agent technology with the self-organizing networks for intrusion detection.

Key-Words: - Intrusion detection system, neural networks, self-organizing maps, visualization.

1 Introduction

Intrusions detections for computer systems are rapidly becoming one of the most important threats for information systems. A major concern is the high rate of false alarms produced by Intrusion Detection Systems which undermine the applicability of such systems. Over the past few years, agent based intrusion detecting systems have emerged as a new solution [2]. Agents represent a new generation of computing systems and are one of the more recent developments in Intrusion Detection Technology. Unlike an expert system, an agent is embedded in its environment. It can dynamically construct new rules as it works, and is capable of using sensors to monitor environment and then take protective actions. This paper describes an agent based system which uses intelligent modules, such as polynomial neural network and self organizing maps for detection of anomalies and intrusions. We propose a framework which provides early warning when attack activities are detected. Fuzzy Agent-Based Intrusion Detection System (FABIDS) provides a framework for integrating data collecting sensors, database, algorithms and agents.

2 Intrusion Detection Systems

In general, there are three categories of Intrusion Detection systems: host based, network based, and application based. This classification depends on the type of sensors they use to collect data in order to detect possible attacks. In the host based approach every host has its own IDS agent and it collects data by monitoring connection attempts to various ports. A network based IDS collects data at the network level. Their sensors and agents are located somewhere in the network and monitor network traffic. The third type processes data from running applications as input. Intrusion Detection Systems are usually not implemented by using just a single concept, but multiple concepts to gather information to detect anomalous behavior in the system. Agents are autonomous software entities that can act independent from other agents and perform different tasks. Agents are applications with predefined goals and run autonomously. They can for example, monitor an environment and issue alerts or start intervention actions based on how they are programmed. In the case of intrusion detection agents can serve as detectives or monitors by recognizing and retrieving data for analysis and develop real-time alerts. Intelligent agent can assists users and acts on their behalf. Agents can automate repetitive tasks, remember events, summarize complex data, learn, and make recommendations. Intelligent agents continuously perform two main functions, which differentiate them from other software programs: they collect data from environment in which they operate and reason to interpret data and suggest actions. Agents can reduce intrusion detection workload by sifting through large amounts of data for evidence gathering. While there are multiple definitions of intelligent agents, their essential characteristic in intrusion detection is that agents are software computing entities that perform intrusion detection tasks autonomously. Agent technology is not a new single technology, but rather the integrated application of a number of concepts tools and technologies. Agent tasks include: carry common intrusion types and pattern to correlate simple alerts; send back correlated alerts; communicate with other agents. Developers normally do not set out to construct an agent but more typically they add new

functionality to existing application. In order to define the characteristics of an agent further and to distinguish them from any other type of program, the following lists attributes of typical agent systems:
Autonomy: being able to carry out tasks independently is the most important feature of an agent.
Purpose: agents perform a set of tasks on behalf of a user or other agents that are explicitly approved and programmed.
Perception: agents need to be able to affect is environment using some type of predefined mechanisms.
Communications: an agent needs to be able to interact with the users and other agents.
Intelligence: an agent needs to be able to interpret monitored events to make appropriate decisions.
Agents represent a new generation of computing systems and are one of the more recent developments in Intrusion Detection Technology.

3 IDS Design Framework

We developed an intrusion detection architecture called Fuzzy Agent-Based Intrusion Detection System (FABIDS). The architecture of this system is presented below:

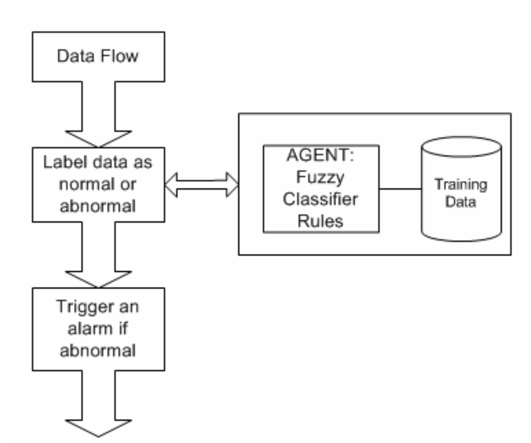
Data Flow
Label data as normal or abnormal
AGENT: Fuzzy Classifier Rules
Training Data
Trigger an alarm if abnormal

Fig.1: FABIDS Architecture

The system consists of multiple intelligent decision support modules, such as fuzzy inference module, classifier, database, etc. This framework integrates several modules such as data collecting sensors, database, fuzzy pattern classification etc [19]. The heart of the controller inference engine is a set of if-then rules whose antecedents and consequences are made up of linguistic variables and associated fuzzy membership functions. Consequences from fired rules are numerically aggregated by fuzzy set union and then defuzzified to yield a single crisp output as the control. Since the differences between the normal and abnormal activities are not distinct, but rather fuzzy, the Fuzzy Inference module can reduce false alarms in determining intrusive activities. For detailed description see [20]. We are processing log files using event correlation engine similar to Risto Vaarandi's powerful Perl event correlation engine described in his article 'A Data Clustering Algorithm for Mining Patterns From Event Logs' published in Proceedings of the 2003 IEEE Workshop on IP Operations and Management [13]. Our algorithm, with similarities to the Apriori and Max-Miner algorithms is implemented in Java. Recently, we have experimented with two new modules: neural networks based and self organizing based.

3.1 Neural network based module

Artificial neural network technology offers a potential solution to intrusion detection problem. Neural networks can perform clustering or categorization. This is also known as unsupervised pattern classification. A solution to a clustering problem shows the similarities between patterns and structures the data so that similar patterns are in the same group. The Self-Organizing Map (SOM) neural network algorithm formulated by Kohonen [3] is good at reducing multidimensional data to fewer dimensions, making it a good solution to the clustering problem. In this paper we investigate the application of self-organization modeling techniques, such as the Group Method of Data Handling (GMDH) in modeling financial systems. A combination of parametric and non-parametric GMDH algorithms is introduced to perform the one step ahead prediction [4] Group Method of Data Handling (GMDH) creates the model that includes only the most influential variables. The GMDH algorithms are based on a sorting-out procedure of model simulation and provide the best model according to the criterion given by the researcher. This model describes relations between their elements and the state of the whole system. Most of GMDH algorithms use polynomial referenced functions.
General connections between input and output variables can be shown by Volterra functional series. A discrete analogue of Volterra series is Kolmogorov-Gabor polynomial

$$y = a_0 + \sum_{i=1}^{N} a_i x_i + \sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij} x_i x_j + \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} a_{ijk} x_i x_j x_k + ...,$$

where y- output variable vector,
$(x_1, x_2, ..., x_N)$
input data,
$(a_1, ..., a_N, ..., a_{ij}, ..., a_{ijk}, ...)$
vector of coefficients or weights.

Input data might consist of independent variables, functional expressions or finite residues. The key feature of GMDH algorithms is a partition of input data into two subsets. The first one is used to compute coefficients of the polynomial using the list square technique and to evaluate internal error by some criterion. The second one is used to calculate external error using information, which is not applied for the coefficients computations.

3.2 SOM based visualization module

Self Organizing Maps are a technique used for finding groups or clusters within a dataset. The difference between using SOMs instead of networks for classification and clustering is that no targets are needed when using SOMs. The SOM will automatically find groups within the data. The main properties of Self-Organizing Maps [3] are as follows: The mapping represents the full set of data in an ordered form. Mutual similarities in the data samples will be represented as geometric relationships on the map. A SOM can be trained on the multi-dimensional input space *(x_1, $x_{2,...}$ x_N)*, and the resulting map used to classify each input. This relies upon the topology of the input space where the similarity of each of the inputs is important:

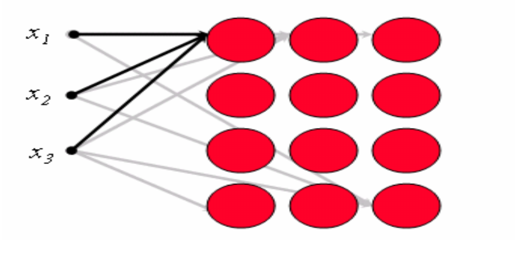

Figure 2: SOM network

The winning neuron for an input to a SOM is defined as:

$$i(x) = \arg\min_j \|x - w_j\|$$

where *i(x)* is the winning neuron index, w_j is the weight vector for neuron *j* and the Euclidean distance metric is used:

$$d = \|x - w\| = \sqrt{\sum_{i=1}^{m} (x_i - w_i)^2}$$

The weights in a SOM are updated at time step *t*+1for neuron *j* using:

$$w_j(t+1) = w_j(t) + \eta(t) h_{j,i}(t)(x - w_j(t))$$

where *η(t)* is the learning rate
and $h_{j,I}(t)$ is the neighborhood for neuron *j*
given the winning neuron *i* at time step *t*.

The structures in the data set can automatically be visualized on the map whereby the degree of clustering is represented by shades of gray (see Fig.3)

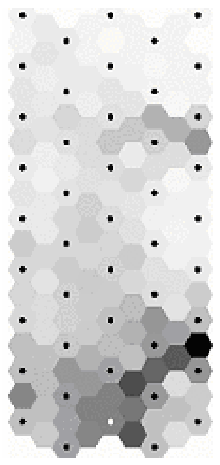

Fig. 3: Visualization of a SOM

4 Experiments

While conducting the research for this paper, the researcher was provided full access to the SNORT logs [19, 20] The basic SNORT architecture is made up of three main parts, the packet decoder, the detection engine and the alerting and logging system. The packet decoder can collect TCP/IP traffic at a blinding rate. Before the engine can compare any of the signatures in its database to the packets, the packet data is passed through a number of user-configurable preprocessors. These preprocessors can reassemble TCP packets into sessions, handle fragmented traffic, and even detect scans and probes. After the preprocessors have formatted the packet data to make it easier to search, the detection engine examines the data for contents that match any of the signatures in its database. If any of the signatures are matched, then the action prescribed for the signature is taken by the third part of SNORT, the alert/log system. If configured, SNORT will also capture the packet data relating to the alert and store it on the hard drive. The alert system will publish alerts to an area on the file system for examination or to a remote analysis console through standard remote log formats like syslog. To encode the descriptions of various attacks a range of positive integers is assigned to each of the attack in the following way.

Entry point (1 bit of information) Web server software (ISAPI filters, Perl modules, etc.) or web application (HTML, server-side and client-side scripts, server components, SQL sentences, etc.)

Vulnerability (3 bits of information) Code Injection, HTML manipulation, Overflows, Misconfiguration (default directories, sample applications, guest accounts, etc.) X if Not applicable,

Threat (3 bits of information): Authentication , Authorization, Confidentiality, Integrity, Availability,

Auditing Action (4 bits of information): Read, Modify, Delete, Fabricate, Impersonate, Bypass, Search, Interrupt, Probe, Unknown,

Length (1 bit of information): Expected, Unexpected (unusually long), X - Not applicable,

HTTP element (7 bits of information): GET/POST, HOST, COOKIE, REFERER, TRANSLATE, SEARCH, PROPFIND

Target (1 bit of information) Web application (source files, customers' data, etc.), Platform (OS command execution, system accounts, network, etc.)

Scope (1 bit of information) Local (one user affected), Universal (all users affected), X - Not applicable

Privileges (1 bit of information), 0 - Unprivileged user, 1 - Administrator/root, X - Not applicable.

Let us consider typical common attacks directed against different types of web servers and platforms:
0, X, 1, 9, 0, 01, 1, X, 0
0, 1, 2, 0, 0, 01, 0, X, X
1, 0, 1, 3, 0, 01, 1, X, 0

Let us explain the last description. The web application allows SQL injection. The attacker exploits this vulnerability by executing a SQL Server extended procedure and adds himself to the OS users. These encoding vectors are useful in a number of ways, especially in intrusion detection systems. An intrusion detection system (IDS) detects and reports attempts to break into or misuse networked computer systems in real time. A traditional IDS consists of three functional components: A monitoring component, such as a packet capturer, which collects traffic data. An inference component, which analyzes the captured data to determine whether it corresponds to normal activity or malicious activity. An alerting component, which generates a response when an attack has been detected. This response can be passive such as writing an entry in an event log or active such as changing configuration rules in the firewall to block the attacker's IP address. Coding web attacks into vectors could helps the post processing of IDS alerts. Encoding web attacks into vectors helps the application-level firewall to decide about the action to be taken when an attack is detected. The most important advantage of this scheme over data compression methods is that the decompression is not needed in the applications. Real world examples of attacks against different platforms, web servers, and applications are given to illustrate how this taxonomy can be applied. In our experiments we also used various sets of benchmark data such as the KDD Cup 1999 Intrusion detection contest data, SNORT logs and real-time data. The KDD Cup 1999 data was prepared by the 1998 DARPA Intrusion Detection Evaluation program by MIT Lincoln Labs. Lincoln labs set up an environment consisting of a local-area network simulating a typical U.S. Air Force LAN. They acquired nine weeks of raw TCPdump data that was processed into connection records. The original data contains 744MB of data with 4.94 million records. The dataset has 41 attributes for each connection record plus one class label specifying one of 24 attacks or normal condition. All these attacks fall into four major categories: Denial of Service (DoS), Remote to User (R2U), User to Root (U2R) and Probing (Probe). While conducting the research for this paper we were provided full access to the SNORT logs from one of the departmental servers [20] The FABIDS sensors were also monitoring network devices: a NAT Router/Firewall and a web server. The NAT Router/Firewall was configured to allow Internet access to the web server, by mapping selected ports to the web server behind the NAT Router/Firewall on the internal network. This configuration was selected to allow a single attack to simultaneously attack the NAT Router/Firewall and the web server so we could generate events logs with identical timestamps to ensure that we could successfully merge data from multiple sensors.

5 Conclusion

As computer attacks become more and more sophisticated, the need to provide effective intrusion detection methods increases. Network-based distributed attacks are especially difficult to detect and require coordination among different intrusion detection components. We propose a solution that responds to such requirements

References:

[1] S. Axelsson. "Intrusion Detection Systems: A Taxonomy and Survey." Technical Report No 99-15, Dept of Computer Engineering, Chalmers University of Technology, Sweden, March 2000

[2] J. Balasubramaniyan, J.O. Garcia-Fernandez, D. Isacoff, E.H. Spafford, and D.M. Zamboni. "An Architecture for Intrusion Detection using Autonomous Agents." Technical Report, Dept. of Computer Science, Purdue Univ., West Lafayette, IN, 1998

[3] T. Kohonen, Self-Organizing Map, 2nd ed., Springer-Verlag, Berlin, 1995.

[4] J.A. Muller, A.G. Ivakhnenko and F. Lemke, GMDH algorithms for complex system modeling, Mathematical and Computer Modeling of Dynamical Systems, vol.4. no.4. pp. 275-316, 1998.

[5] L. Anastasakis and N. Mort, The development of self-organization techniques in modeling: a review of the group method of data handling (GMDH), ACSE Research Report No 813, University of Sheffield, UK, 2001 or www.shef.ac.uk/acse/research/students/l.anastasakis/813.pdf

[6]T. Kohonen, S. Kasaki, K. Lagus, et.al. Self-Organization of a massive document collection. IEEE transaction of Neural Networks 2000, V.11, 3, pp.574-585.

[7] H.R. Madala and A.G. Ivakhnenko, Inductive Learning Algorithms for Complex Systems Modeling, Boca Raton: CRC Inc., 1994.

[8] A.G. Ivakhnenko, Polynomial theory of complex systems, IEEE Trans. Syst. Man Cybern. SMC-1 (1971) 364–378.

[9] S.J. Farlow (Ed.), Self-organizing Method in Modeling: GMDH Type Algorithm, Marcel Dekker, New York, 1984.

[10] Kohonen, Teuvo. The Self-Organizing Map. Proceedings of the IEEE. Vol. 78.n0, Sept 1990. p 1464-1480.

[11] Kasabov Nikola. Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering. MIT Press. Cambridge Massachusetts. 1996.

[12] Teuvo Kohonen, Erkki Oja, Olli Simula, Ari Visa, and Jari Kangas. Engineering Applications of the Self-Organizing Map. Proceedings of the IEEE Vol. 884 no 10 October 1996.

[13]Risto Vaarandi, "SEC - a Lightweight Event Correlation Tool", Proceedings of the 2nd IEEE Workshop on IP Operations and Management, 2002.

[14]D. Bulatovic and D. Velasevic, "A Distributed Intrusion Detection System Based on Bayesian Alarm Networks," In Proc. of CQRE'99, LNCS 1740, pp. 219–228, 1999.

[15]J. Cannady. "Artificial Neural Networks for Misuse Detection." In Proc. of the 21st National Information Systems Security Conf., VA, 1998, pp. 441-454

[16] C.A. Carver, J.M. Hill, J.R. Surdu, and U.W. Pooch. "A Methodology for using Intelligent Agents to Provide Automated Intrusion Response." In Proc. of the IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop, West Point, NY, 2000

[17] Richard E. Neapolitan. Probabilistic Reasoning in Expert Systems. Wiley, 1990.

[18] S. Northcutt*, Network Intrusion Detection: An Analyst's Handbook*, New Riders, 1999

[19]Wasniowski R A, Intrusion Detection System with Fuzzy Logic Agent, RAW-TR-01-09

[20]Wasniowski RA, Agent Based Design Methodology, RAW-TR-00-12

[21]Wooldridge, M., and Jennings, N. (1995) "Intelligent Agents: Theory and Practice," Knowledge Engineering Review, Vol. 10, No. 2.