An Agent-Based Knowledge System for Intrusion Detection

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

Abstract: - In this paper we propose a framework for intrusion detection called Fuzzy Agent-Based Intrusion Detection System. A unique feature of this model is that the agent uses the fuzzy logic to process log files. This reduces the overhead in a distributed intrusion detection system. We have developed an agent communication architecture that provides a prototype implementation. This paper discusses the issues of combining intelligent agent technology with the intrusion detection domain.

Key-Words: - intrusion detection, fuzzy logic, agents, computer network security, Multi-Agent System, Knowledge System

1 Introduction

Web attacks are rapidly becoming one of the fundamental threats for information systems connected to the Internet. When the attacks are analyzed it is observed that most of them are similar, using a reduced number of attacking techniques. It is generally agreed that classification can help designers and programmers to better understand attacks and build more secure detection and response applications. Over the past few years, intrusion detecting agents have emerged as a new software solution. Agents represent a new generation of computing systems and are one of the more recent developments in Intrusion Detection Technology. 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 a new single technology, but rather the integrated application of a number of concepts tools and

technologies. 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 reason through simple to elaborate networks of rules:

IF X AND Y THEN Z.

To develop intelligence in agents, certain steps can be taken. They involve the following type of rule sequencing and construction: The user or developer provides a set of rules that describe a desired behavior: When X and Y happens, then do Z. The reasoning system is provided with interfaces to perform or initiate various desired actions; for example, it may require that an alert be made by sending a message to a system object, by writing a file, or by other system action that a program can perform. After the reasoning system is initiated, it

can wait for an event to arrive. It will extract facts from the event and then evaluate its rules to see if the new facts cause any of them to fire. If one or more rules fire, it may cause additional action to be initiated or a record to be written or updated. The above process leads to the creation and use of conditional rules and logic, which can be coded in a variety of ways.
Here is an example:

IF (Condition 1)
OR (Condition 2)
AND (Condition 3)
THEN (Action)

Unlike an expert system, an agent is embedded in its environment and can perceive and react to it using inputs of conditions. It can dynamically construct new rules as it works; in other words agents are capable of using sensors to monitor their environment, develop new rules, and then take actions independently.

2 Intrusion Detection Problem

A fuzzy set may be represented by a mathematical formulation known as a membership function. That is, associated with a given linguistic variable are linguistic values or fuzzy subsets expressed as membership functions which represent uncertainty or imprecision in values of the linguistic variable. These functions assign a numerical degree of membership to a crisp (precise) number. More precisely, over a given universe of discourse X, the membership function of a fuzzy set, denoted by $\mu(x)$, maps elements x ? X into a numerical value in the closed unit interval, i.e. $\mu(x)$: X ? [0, 1].Implementation of a fuzzy system requires assigning membership functions for inputs and outputs. Inputs to a fuzzy system are usually measured variables, associated with the state of the controlled object, that are fuzzified before being processed by an inference engine. 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. For detailed introductions to fuzzy set operations, and concepts of fuzzification, inference, aggregation, and defuzzification see [26,27]

3 IDS Design methodology

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

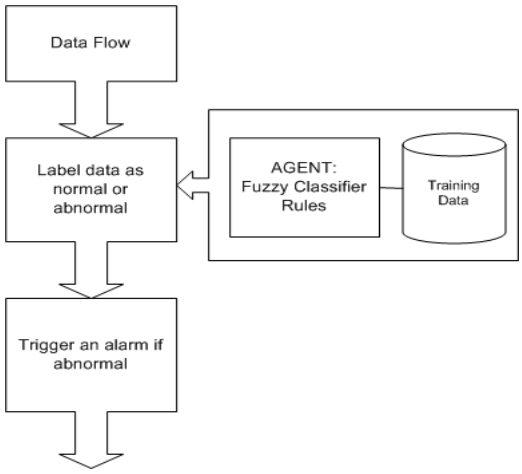

Fig. System Architecture

Agent software packages are readily available from various sources, including the Web, from such sites as agentland.com. Using an agent, as opposed to a search engine, has the advantage that all of these links can be viewed at any time. The advantages of this architecture is that a low volume of data must be sent over network in a distributed intrusion detection scenario This feature allows easy exploration of the trade-off between sensitivity and selectivity that affects the rate of false decisions. The distributed nature of the system and the fact that each agent is an autonomous entity increases the efficiency of the processing.

A MAS (multi agent systems) is an emerging subfield of AI that aims to provide both principles for construction of complex systems involving multiple agents and mechanisms for coordination of independent agents' behaviors. While there is no generally accepted definition of "agent" in AI [16], for the purposes of this study, we consider an agent to be a software entity. A MAS allows the subproblems of a constraint satisfaction problem to be subcontracted to different problem solving agents with their own interests and goals. An ontology is a formal, explicit specification of a shared conceptualization. An ontology can be viewed as a document or file that formally defines the relations among terms. The most typical kind of ontology is defined in terms of a taxonomy of terms and a set of inference rules.

Any Knowledge Based System consists of at least two fundamental parts: domain knowledge and problem-solving knowledge. Ontologies mainly play a role in analyzing, modeling and implementing the domain knowledge [24].

The purpose of using ontologies iis to enable knowledge sharing among agents. For example, the action of "add new rule" will evoke the same feature set, in terms of structure and behavior, in both the sender agent and the receiver agent.
So agents can understand the structure of the system and the meaning of agent actions. In figure 2, it shows the five actions: "ADD", "ROMOVE", "REPLACE", and "SET TIME".

Agent action type	Corresponding operations
ADD	Adding a new rule
REMOVE	Removing rule
REPLA CE	Removing the former rule and adding a new one
SET TIME	Set or reset the total time

Fig 2. Agent action types

A key feature in then is to specify the common ontology in a representation that subsumes the models for individual agents. We adopt Protégé 2000, developed by Stanford University, to generate form-based interfaces that could check for constraints violation. There are two steps to establish an ontology. Firstly, we get the "bean generator" for Protégé from web site. With the "beangenerator", it can be used to generate java files representing an ontology that is used for with the JADE (Java Agent DEvelopment framework) environment. Thus, based on these standardization and ontologies, the agent implementation in the JADE by Bellifemine [3] can be applied here.
The "beangenerator" is designed as a plug-in for Protégé. Secondly, the process of generating is easy since programmers are required to enter some related concepts, and their slots with data type.
Agent Communication: agent will not move within different hosts. Agents communicate with each other by sending alert information via Messages. Agent task:
1. carry common intrusion types and attern to correlate simple alerts
2. send back correlated alerts to central console
3. communicate with agents

4 Experiments

While conducting the research for this paper, the researcher was provided full access to the SNORT logs [26,27] 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 pre-processors. These pre-processors 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.

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 or systems. We propose a solution that is more effective than current IDS's. This architecture allows local analysis and sharing of results as well as minimizing the communication costs.

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] Bellifemine, F., Caire, G., Trucco, T., & Rimassa G. (2000). JADE Programmer's Guide. Retrieved October 12, 2001 from World Wide Web: http://sharon.cselt.it/projects/jade/.

[4] Bloemeke, Mark and Marco Valtorta. "The Rumor Problem in Multiagent Systems." USC CSCE TR-2002-006, Department of Computer Science and Engineering, University of South Carolina, Columbia, 2002

[5] 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.

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

[7] 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

[8] Cowell, Robert G., A. Philip Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer-Verlag, 1999

[9] William DuMouchel. "Computer Intrusion Detection Based on Bayes Factors for Comparing Command Transition Probabilities," Technical Report No. 91, Feb 99, National Institute of Statistical Sciences.

[10] Caglayan, A. and Harrison, C. (1997) *Agent Sourcebook*, New York: Wiley Computer Publishing.

[11] Franklin, S. and Graesser, A. (1996) "Is It an Agent, or just a Program?: A Taxonomy for Autonomous Agents," *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, Springer-Verlag.

[12] Kotz, D. and Gray, B. (May 1, 1999) "Mobile Agents and the Future of the Internet," *Workshop Autonomous Agents, Seattle, WA.*

[13] Vaibhav Gowadia, Csilla Farkas, and Marco Valtorta. "Intrusion Analysis with Soft Evidential Updates," USC CSCE TR-2001-005, Department of Computer Science, University of South Carolina, Columbia, 2002.

[14] G. Helmer, J. Wong, V. Honavar, and L. Miller. "Lightweight Agents for Intrusion Detection." *Submitted to Journal of Systems and Software*

[15] Finn V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001.

[16] Russell, S. J. & Norvig, P.(1995). Artificial Intelligence—A modern approach. Upper saddle River ,NJ:Prentice Hall Inc.

[17] W. Jansen, P. Mell, T. Karygiannis, and D. Marks. "Applying mobile agents to intrusion detection and response." NISTIR -6416, September 1999

[18] Young-Gyun Kim, M. Valtorta, and J. Vomlel. "A Prototypical System for Soft Evidential Update." USC CSC E TR2002-005, Department of Computer Science and Engineering, University of South Carolina, Columbia, 2002.

[19] Steffen L. Lauritzen and David J. Spiegelhalter. "Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems." *Journal of the Royal Statistical Society*, Series B, 50 (1988), 2, pp.157-224.

[20] W. Lee and S.J. Stolfo. "Data Mining Approaches for Intrusion Detection." *In Proc. of the 7th USENIX Security Symp*, San Antonio, TX, 1998, pp.79-94

[21] M. Meneganti, F.S. Saviello, and R.Tagliaferri. "Fuzzy Neural Networks for Classification and Detection of Anomalies." *IEEE Trans. On Neural Networks*, 9/5, 1998, pp. 848-861

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

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

[24] J. Moy. OSPF version 2. Internet Draft, RFC -2178, July 1997

[23] Judea Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan-Kaufmann, 1988.

[24] Studer, R., Benjamins, V. R., Fensel, D. (1998). Knowledge Engineering: Principles and Methods. Data Knowledge Engineering, 25 (1-2).

[25] Marco Valtorta, Young-Gyun Kim, and Jirí Vomlel. "Soft Evidential Update for Probabilistic Multiagent Systems." *International Journal of Approximate Reasoning*, 29, 1 (January 2002), pp.71-106.

[26] A. Valdes and K. Skinner. "Adaptive, Model-Based Monitoring for Cyber Attack Detection." *In Proc. RAID*, 2000, pp. 80-92

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

[28] Wasniowski RA, Intrusion Detection System with Fuzzy Logic Agent, RAW-TR-01-09

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