Multiagent Network Security System using FIPA-OS

Taraka D. Peddireddy; University of South Carolina; Columbia; South Carolina
Jose M. Vidal, Assistant Professor, University of South Carolina, Columbia, South Carolina

Keywords: Multiagent Network Security, Distributed Systems Security, Distributed Intrusion Detection Systems

ABSTRACT

This paper describes a security framework in distributed systems where an Intelligent Agent handles the security monitoring at each host. The agents are made responsible for alerting the system administrators about an attempted intrusion or misuse for a particular system. Recently, there has been an increase in the number of reports of the attacks, which are wide spread across the network and affecting a chain of systems before they attack the actual target system. To detect such attacks, the amount of information associated within a single isolated system is inadequate for an agent to confirm an intrusion. Therefore, the need for a framework that allows the agents to negotiate with their co-agents to share information about an intrusion, thereby aiding in effective handling of Intrusion Detection is emphasized. Our design aims at developing such a framework in the FIPA-OS (Foundation for Intelligent Physical Agents – Open Source) environment, which provides most of the source code for building agents on its platform. The concept of mutual co-operation among agents has been developed as a means of queries. These queries are carried out by tasks associated with each agent. The protocols to support these interactions by means of queries are explained. The issues and requirements involved in standardizing formats, interaction protocols and architectures to co-manage intrusion detection are discussed.

1. INTRODUCTION

Security and privacy are the growing concerns in the open distributed software systems due to Internet's rapid growth and the desire to conduct business over it safely. This desire has led to the advent of many security architectures and protocols, which deals with authentication, cryptography, and authorization to avoid a possible intrusion. There has been significant work in the field of intrusion detection that comes into picture after an attack. Most of the projects are largely focused on the analysis of attacks within a single isolated system. Incidents, however, often consist of a large series of widely distributed exploits, involving numerous systems, networks, operating systems and applications. Intruders often compromise multiple systems when they attack a target site. At each compromised system, there may be signs of intrusive activities that agents of the respective systems discover. By gathering information from those systems (from agents of those systems), we can determine the widespread nature of attacks against our networked systems. It is also possible that our systems may have been compromised and are serving as unwitting participants in large-scale attacks against several sites. External contacts assist us in security monitoring, greatly extending our ability to detect intrusions. Therefore, the need arises for systems to co-operate with each other, to manage such diverse attacks across networks and time. However, in co-operative situations, trust is an important issue in most of the decisions. It is not always possible that hosts on distributed systems reveal the information about an intrusion, out of fear of bad reputation or of leaking sensitive information. Hence, the purpose of this research study is to standardize the information formats and automate the interaction protocols for effective communication among untrustworthy and self-interested agents.

2. PROJECT DESCRIPTION

2.1 Related Work

A number of efforts have been made to come up with distributed IDS. In Distributed Intrusion Detection System (DIDS [19]) information is kept strictly centralized, and no agent technology is used. Data from all remote sources come to one place for processing. In JAM project [9], local fraud detection agents provide Intrusion Detection within a single system, JAM also supports Meta learning system that combines the collective knowledge acquired by individual local agents. In Common Intrusion Detection Framework (CIDF [3]), the Common Intrusion Specification Language (CISL) is designed to express information about intrusions. The architecture defines relationships between event generators, event analyzers, event databases and response units but no mention is made of agent technology as such in this framework. A recent article about Biological Network Security on SecurityFocus.com [5] discusses the need for open and common mechanism for communicating among different security mechanisms. This project derived motivation from the above projects and articles to extend the current research on Distributed Intrusion Detection systems.

2.2 Theoretical Model

This application provides the capability to collect information from co-hosts through agents, with varying trust levels. This ability is demonstrated by building a simulation of a system where an agent represents each host. Distributed attacks often leave trace information in log files and audit files, files and processes left behind by an intruder. This trace information is used to search for suspicious events or connections that require further investigation. There are software packages (like TCP/IP daemon wrapper package [9]), which inspect the logging information and detect signs of intrusion. Once alerted by the intrusion detection software that an intrusion has been detected, we need to analyze that intrusion by investigating to what extent our systems or data have been compromised. We then respond to that intrusion based on the results of the analysis.

This analysis is carried out by the respective host agents for finding the information like

- What attacks are used to gain access?
- What systems and data did an intruder access?
- What an intruder did after obtaining access?

To perform such analysis, agents create, use and maintain a list comprising of a sequence of the agents to contact, and other procedures for informing co-agents quickly based on the type of intrusion. As each intrusion differs from each other in many ways namely, how it has happened and from where it came from, agents conduct different analysis mechanisms to deal with different types of intrusions. During its analysis, it is important to keep communication with other agents about an intrusion. If they are experiencing unexpected behavior by an intruder, we may gain some information that will help us to protect our own systems. Agents may be notified by other agents about the evidence of attacks against our systems originating from their systems and vice versa. This implies that our systems may have been compromised by the intruder to hide his or her tracks and launch an attack against other systems.

By receiving such notifications our agent will further investigate it and alert us if it finds any confirmed intrusion. During its investigation it may contact some other host agents involved in that intrusion or need to be informed about the intrusion. Consequently, the agent ends up in sharing information among many agents for single intrusion detection. While doing so, agent should take care of sharing information only on a need-to-know basis, and based on trust levels among agents, sanitize sensitive information, if required.

3. SYSTEM DESIGN

I have designed a simulation of security framework that provides a service of retrieving the information from a distrusted network. This information further helps to detect intrusions on hosts. The information retrieval at a host is achieved by the agent residing at that host. The architecture of the project, which better illustrates this idea is shown in Figure2. In the context of our application, an agent is defined as an encapsulated software entity with its own state, behavior and thread of control and ability to interact and communicate with other entities – including people, other agents and systems. An agent is autonomous in its action and communicates with other agents using an agent communication language like FIPA-ACL / KQML based on speech acts. The work is being carried out in the FIPA-OS environment.

3.1 FIPA-OS Description

FIPA Open Source [16] is an open agent platform, originating from Nortel Networks.

The platform supports communication between multiple agents using an agent communication language that confirms to the FIPA (Foundation for Intelligent Physical Agents [14]) agent standards. The reason for choosing fipa-os platform is because agents do certainly reside on multiple platforms and exhibit different behaviors. FIPA-OS can interoperate with other heterogeneous FIPA complaint platforms (e.g.: JADE [2]). There is a large scope for future enhancements to this project by making the agents follow the FIPA standards.

FIPA-OS has built-in support for:

- Different types of agent shells for producing agents, which can then communicate with each other using the fipa-os facilities.
- Multi-layered support for agent communication
- Message and Conversation Management
- Dynamic platform configuration

FIPA-OS is designed to operate in a heterogeneous open source environment and supports multiple encodings for the content. It also supports multiple transports such as IIOP (using a variety of CORBA API's), RMI and TCP.

3.2 Methodology

We assume that log files of every host are inspected by some software mechanisms at some predetermined time interval and any discovered unusual entries are documented. Some of the unusual entries associated with respective log files are shown in Figure 1.

The primary aim is to find out whether these unusual entries lead to any confirmed intrusion for which additional information about that unusual entry is needed. There are many kinds of intrusions characterized, in the way the intrusion is performed particularly the way they occur, and how they affect the target host etc. Typically intrusions are classified by the sequence of events that lead to that particular intrusion.

One such classification of intrusions is considered, which include some well-known attacks and a comprehensive set of events associated with those intrusions. Section 3.3 is a listing of suspicious events associated with each particular type of intrusion.

3.3 Classification of Intrusions

The project focuses on automating the detection of the following subset of attacks.

Invalid Logins/ Suspicious Logins

- Logins not logged for an abnormal length of Time
- Logins at Unusual times
- Short Login times
- Logins from unexpected locations
- Failed login attempts

Illegal Connections/ Abnormal Connections (from / to)

- Connections from/to Unusual Locations
- Half open connections
- Sudden spike in network traffic
- Telnet connections without output from w or who commands

Unusual Processes / Suspicious Processes

- Processes that take a long time
- Processes with unusual start times
- Processes with high % of CPU time (a sniffer)
- Processes without a controlling terminal
- Processes with unusual names
- Large number of processes at a time

Suspicious files/ Unauthorized modification of files

- Unexpected changes to password files or access control lists
- Unexpected size of file (may be a Trojan)
- Unfamiliar files
- System files that appear to have been modified recently
- Short systems files indicating that this file has been edited or deleted

Type of Log file	Unusual Entries
User activity	• Repeated failed logins • Logins from unusual locations • Unusual processes run by the user • Unauthorized accesses to files • Logins at unusual times
Network connections	• Connections from unusual locations • Connections to unusual locations
Web server activity	• Repeated attempts to misuse the server • Flooding activities that could cause a denial of service problem
Network Traffic monitoring	• Half open connections
Systems activity	• Unexpected shutdowns • Unexpected reboots

Figure 1 . Table of Unusual entries within log files

Denial of service

- Flooding / ICMP bombing
- Email Bombing
- Smurf / Syslog / SNMP bombs

Alteration of System Privileges

- Alteration to su, setuid, setgid files to change the authentication status

Protocol violations

- Invalid bits in a TCP packet
- Unusual port combinations in TCP and UDP packets (e.g.: Teardrop, Ping of Death)

3.4 Process of Implementation

Given the suspicious entry, we need to analyze the abnormalities associated with that entry and check whether it belongs to those sequences of events that result in an intrusion. Agents read the unusual entries documented in the database by some pre-selected software and start its investigation about the abnormalities. The investigation may lead to gathering information from a sequence of queries posed to respective agents.

The agent formulates the following type of queries, with respect to the suspicious entry in the log file:

Can it be explained by an authorized user?

Can it be explained by known System activity?

Can it be explained by authorized changes to programs?

Each query is handled by a reusable Task class, which is developed independent of the agents using that task and type of intrusion task it is dealing with. Agents are provided with a variety of tasks, the choice of which will depend on the next query to be handled in the sequence. The task tackles the query by initiating a number of conversations with other agents and getting their responses back to the agent. Based on the responses, agent may choose to further call another task for the next query or may end up determining that no further investigation is required. Conversations are built using FIPA performatives.

A typical example of a conversation between two agents A and B can be:

Query-If (Sender: Agent-A
Receiver: Agent-B
Content: (Inform-If
SourceIP Address: 252.23.24.20
DestinationIP Address: 219.29.27.28
Date/Time: 19 SEPT 2001
Username: Paul
Content: *Is user authorized*?)
Reply-with: 0001
Language: FIPA)

Inform-If (Sender: Agent-B
Receiver: Agent-A
Content:
(SourceIP Address: 252.23.24.20
DestinationIP Address: 219.29.27.28
Date/Time: 19 SEPT 2001
Username: Paul
Content: Authorized/ Not authorized / Not Understood)
In-reply to: 0001
Language: FIPA)

The decision tree shown in Figure 3 illustrates an example where Agent A has the goal of finding information about a suspicious login event. Each node in the tree represents a task specifying the information query being handled by the respective agent. Agents have access to the information gathering needs associated with different steps of the task.

Based on this knowledge, the agents decide how to decompose the tasks, what information is needed at each decision point, and when to initiate conversations with other agents to get that information. Information gathering activities associated with a

particular event are automatically activated by models of the task.

From the various responses obtained from the queries that are handled by the predefined tasks, agent gathers sufficient information for classifying the data to determine any occurrence of suspicious activity. At last all confirmed evidences of intrusion, attempted intrusion or misuse are reported to an Internal Security Point of Contact. The user interface provided with this application shows the operations and status reported by the agents of the system. It provides a mechanism for feeding input as a specific type of intrusion making an agent work for that intrusion.

4. SUMMARY AND FUTURE WORK

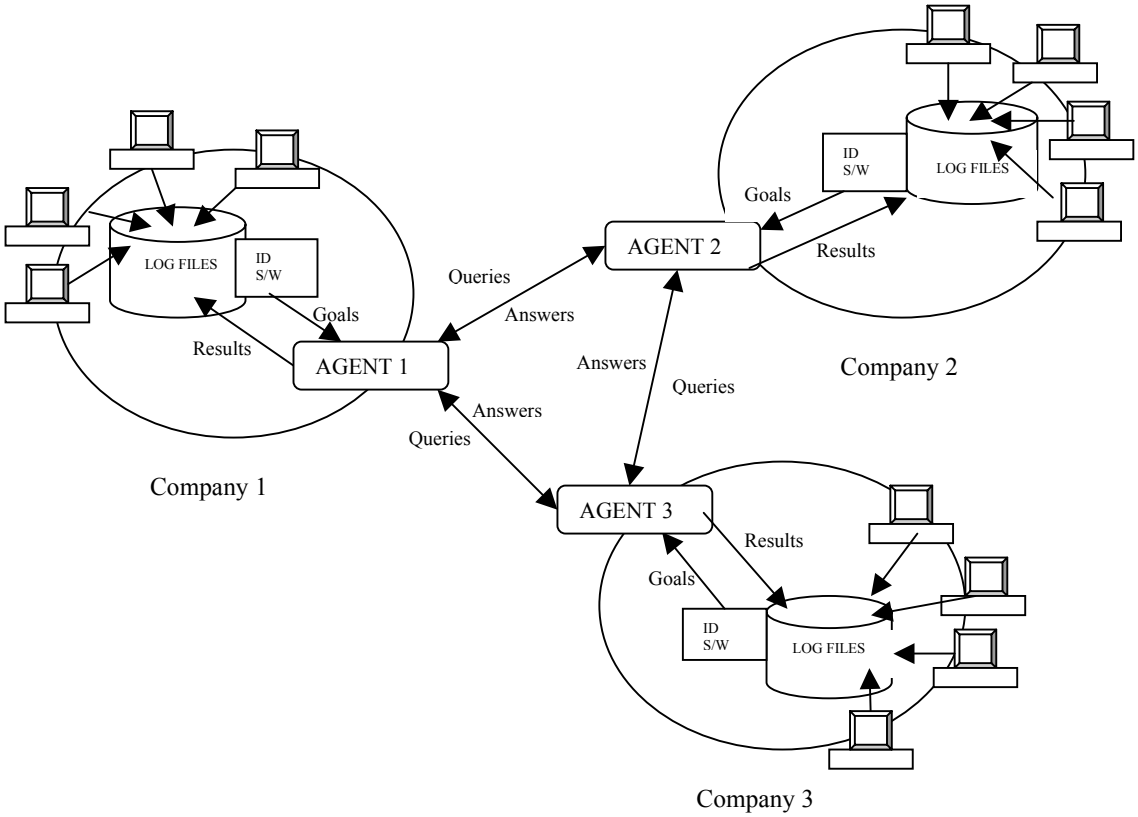
Because this framework provides a temporal view of the knowledge and activity of the monitored distributed system, we believe this system could help system administrators to identify new attacks, spot and defend known attacks, develop better protection and countermeasures for their system.

This paper is an attempt in a step towards the development of specifications for an agent communication language in Distributed systems. The long term goal of this project is to implement and extend the protocols which ultimately evolve into a common language for Distributed Intrusion Detection Systems. This would eventually leads to have a common and open standard for security systems to inter-operate on an Internet wide scale.

5. ACKNOWLEDGEMENTS

I would like to thank my major professor Dr. Jose M Vidal for his constant guidance, support and assistance through out this project.

ARCHITECTURE OF THE PROJECT



ID S/W = Intrusion Detection Software
Goals = Intrusion Specifications - Carried out by Tasks within the Agent
Results = Information about the abnormalities associated with the intrusion

Figure 2. Representation of the Project

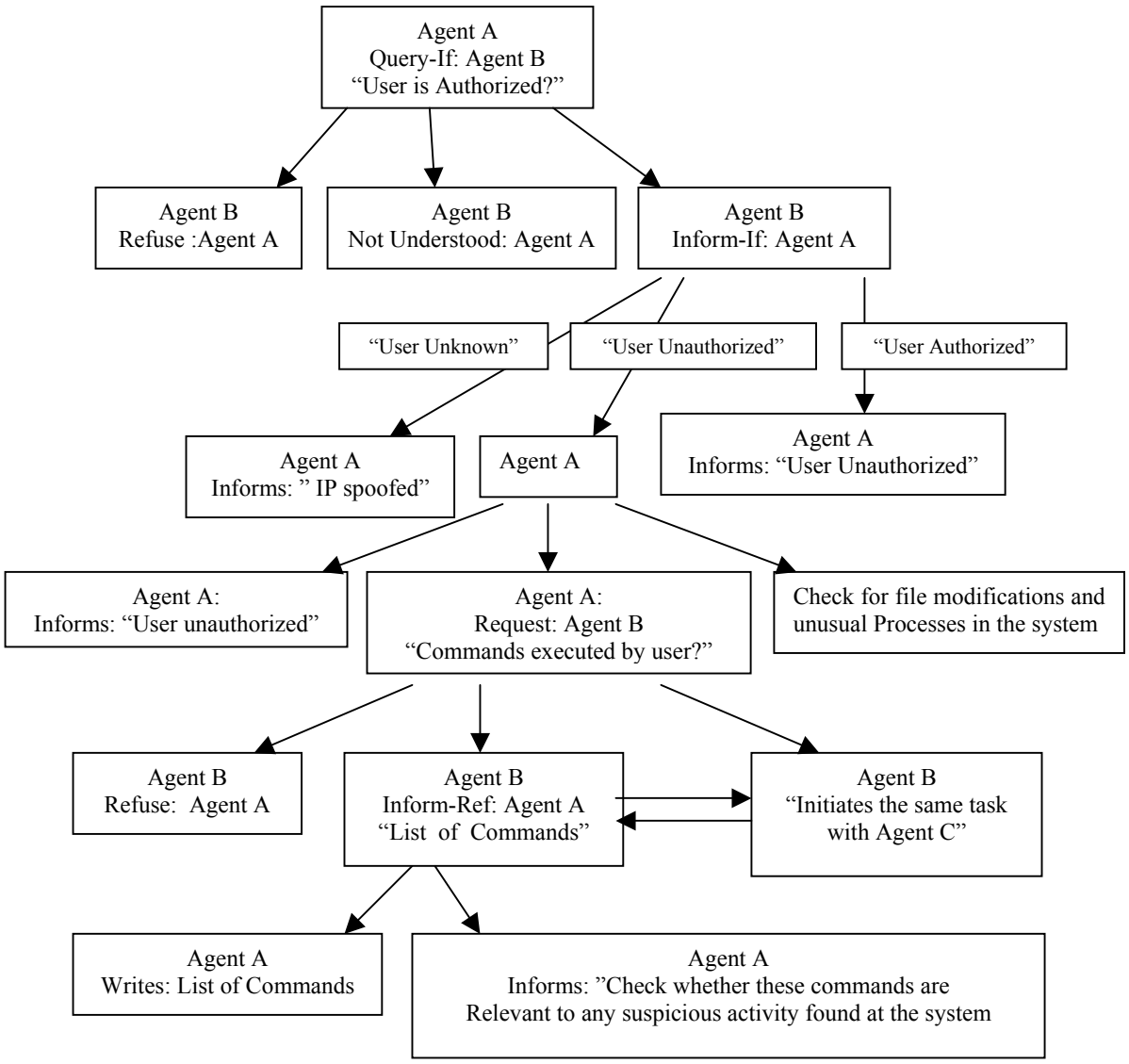


Figure 3. Model of a Task that implements Suspicious Login Protocol

REFERENCES

1. S.Poslad, P. Buckle and R.Hadingham, "The FIPA-OS agent platform: Open Source for Open Standards" , *published at PAAM2000*, Machestor, UK, April 2000.
2. F. Bellifemine, G. Rimassa, and A. Poggi, "JADE - A FIPA-compliant gent Framework", *In Proceedings of the 4th International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents*, London, UK, Dec. 1999, pp. 97-108.
3. S. Staniford-Chen, B. Tung, P. Porras, C. Kahn, D. Schnackenberg, R. Feiertag and M. Stillma, "The Common Intrusion Detection Framework and Data Formats" , March 1998.
4. M. Asaka, S. Okazawa, A. Taguchi and S. Goto, "A Method of Tracing Intruders by Use of Mobile Agent", *.INET99*, June 1999.
5. http://online.securityfocus.com/guest/10094, January 2002
6. K. Kindaley, *A Database of computer attacks for the evaluation of Intrusion Detection systems*, MIIT, May 21 , 1999.
7. K. Sycara, K. Decker, A. Pannu, M. Williamson and D. Zeng "Distributed Intelligent Agents" *,IEEE Expert,* Dec 1996.
8. S. Kumar, "Classification and Detection of Computer Intrusions", PhD thesis, Purdue University, August 1995.
9. http://www.cs.columbia.edu/~sal/JAM/PROJECT/, June 2001.
10. R. Bejtlich, "Interpreting Network Traffic: A Network Intrusion Detector's Look at Suspicious Events", v 2.8 , May 2000.
11. J. P. Anderson, "Computer Security threat monitoring and surveillance", *Technical Report*, James P. Anderson Co., Fort Washington, PA, April 1980.
12. G. G. Helmer., J. S. K. Wong, V. Honavar, and L. Miller, "Intelligent Agents for Intrusion Detection", *Proc. IEEE Information Technology Congerence*, Syracuse, NT, Sept. 1998, pp. 121-124.
13. T. Oates, M.V. Nagendra Prasad and V.R. Lesser, "Cooperative Information Gathering: A Distributed Problem Solving Approach ", *UMASS Technical Report 94-66*, University of Massachusetts, Amherst, MA, Sept. 1994.
14. http://www.fipa.org, June 2001.
15. http://www.cert.org/, December 2001.
16. http://fipa-os.sourceforge.net/, June 2001
17. M. Slagell, "The Design and Implementation of MAIDS (Mobile Agents for Intrusion Detection System)", *Masters Creative Component paper*, Iowa State University, May 2001.
18. http://www.cerias.purdue.edu/homes/aafid/, CERIAS Autonomous Agents For Intrusion Detection Group, December 2001
19. S. Snapp, J. Brentano, G. Dias, T. Goan, L. Heberlein, C. Ho, K. Levitt and B. Mukherjee , "A System for Distributed Intrusion Detection", *COMPCON Spring '91 Digest of Papers*, San Francisco, CA, March 1991, pp. 170-176.