A Prototype MultiAgent Network Security System

Taraka Pedireddy
University of South Carolina
295 Barnes Blvd.
Rockledge, FL, 32955
taraka_reddy@hotmail.com

José M. Vidal
University of South Carolina
Computer Science and Engineering
Columbia, SC, 29208
vidal@sc.edu

ABSTRACT

Distributed Internet-based attacks on computer systems are becoming more prevalent. These attacks usually employ some form of automation and involve the compromise of many systems across the Internet; systems which are not necessarily owned by the same company or individual. The information needed to detect and neutralize these attacks is spread across many machines. A system administrator who wishes to detect and handle these distributed attacks must constantly monitor his systems and communicate with other administrators around the world-a challenging task. In this paper we present our design and implementation of a multi-agent system, built using FIPA-OS, in which agents responsible for different network realms communicate with each other in order to determine if certain suspicious events are actually part of a distributed attack, and to warn each other about possible threats.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*multiagent systems*

General Terms

Design, Experimentation, Languages

Keywords

Distributed security, multiagent systems, FIPA

1. INTRODUCTION

Security and privacy are growing concerns in the open distributed software systems community because of the Internet's rapid growth and the desire for secure transactions over it. This desire has led to the advent of many security architectures and protocols which deal with authentication, cryptography, and authorization. One of the biggest risks to Internet survivability is the growing number of distributed and automated attacks by malicious intruders. The security industry has so far concentrated solely on the development of automated security programs that analyze the attacks within a single isolated system. These programs never use the Internet as a communication medium except when downloading updates from the central server. Security consortiums, on the other hand, concentrate on the publication of security alerts aimed at system administrators. None of these approaches manages to leverage the distributed automated nature of the Internet to serve as a vehicle for its own survival. Meanwhile more and more security incidents 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 nature of attacks against our networked systems. It is also possible that a system may have been compromised and is serving as unwitting participant 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 cooperate with each other in order to manage such diverse attacks across networks and time.

Our design and implementation aims at developing a framework where an intelligent and co-operative agent communicates with the agents in other domains to share information about an intrusion. As such, our system automates the task of distributed intrusion detection while minimizing the amount of agent communication and human intervention needed. The implemented system demonstrates that this type of approach is viable.

2. ARCHITECTURE

Our system implements the capability to collect intrusion specific information from co-hosts through agents. Distributed attacks often leave trace information in log files, audit 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 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 host agents and tries to answer the following questions: Which attacks are used to gain access? Which systems and data did the intruder access? What did the intruder do after obtaining access?

We have designed and implemented an event-based se-

AAMAS'03, July 14–18, 2003, Melbourne, Australia.
ACM 1-58113-683-8/03/0007.

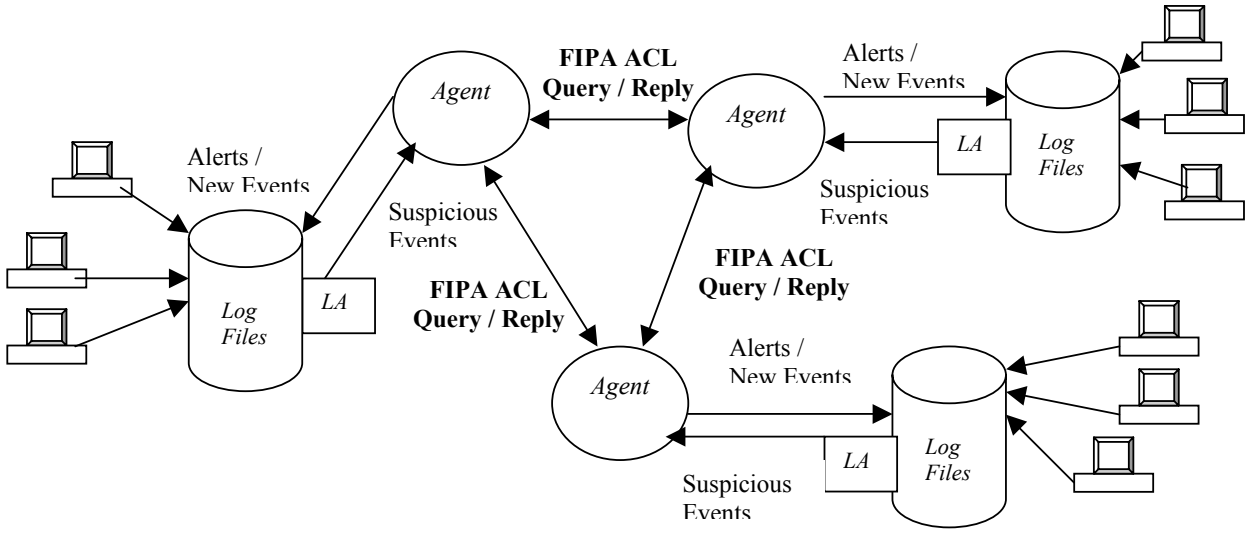

Figure 1: Architecture of the current prototype system. Log files are analyzed by Log Analyzer(LA) program. Any suspicious events become goals for the agent to act on. The responses received out of conversations with other agents are stored back as alerts or new events in the database.

curity framework that provides a service of retrieving the information from a distributed network. This information is then used to detect intrusions on hosts. The architecture of the system is shown in Figure 1. In our framework each domain is represented by an agent. All computers in a domain submit their log files to a central database. We assume that these log files are inspected by a log analyzer program which reports any suspicious events to the agent in its domain. Upon receiving suspicious events, an agent starts its analysis by actively communicating with agents of other domains. The results of the conversations with other agents are stored back as alerts or new events in the database.

In the context of our application an agent is defined as an encapsulated software entity with its own state, behavior, 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 FIPA-ACL. Our agents are implemented using FIPA-OS.

Each suspicious event is handled by a reusable Task class which is developed independent of the agents using that task and type of intrusion the agents are meant to deal with. Agents are provided with a variety of tasks. Agents gather information from other agents by invoking appropriate tasks. They read an event documented in the database and pass it to all the registered tasks. The tasks which can handle this event get executed dynamically. A Task handles the event by initiating a number of conversations with other agents and updates the event using the responses of those agents. In the process, it may receive new alerts or events. Each Task is meant to follow an interaction protocol to deal with a specific type of event. Based on the results returned by the tasks, an agent may choose to invoke another Task or may conclude that no further investigation is required. Conversations are instantiations of interaction protocols, built using FIPA communicative acts. The content of the conversation is expressed in XML.

We have implemented a set of interaction protocols. For example, the alert protocol sends an alert message to a set of machines. The denial of service protocol handles DoS attacks (such as the Mitnick Attack which uses SYN flooding) by allowing the agent that is under SYN flooding to alert other agents to the fact that there is a possibility that an attacker could be pretending to be the one of the agent's machines. In this way, the protocol helps prevent IP spoofing. The suspicious login protocol alerts other agents about suspicious or unusual logins from their machines. This protocol was designed to counteract the DoorKnob attack where an attacker gains illegitimate access to one of the systems and then tries to parlay that into access to other systems.

3. TEST RESULTS

We conducted several preliminary tests on our current framework and successfully derived experimental results. The log files were stored in an ORACLE database. The test cases include tests of all the suspicious events identified so far, scalability, timeouts, and error handling. In general our tests have shown that the number of messages sent grows roughly linearly with the number of agents. Specifically we ran tests with 5, 50 and 150 agents which resulted at most in 5, 50 and 150 messages respectively. This confirms that the agent communication is linear. The execution of each complete protocol was fairly instantaneous.

Our system is a first step towards the development of open security interaction protocols using an agent communication language among distributed intrusion detection systems. The long-term goal of this project is the development of standardized languages and interaction protocols for an Internet-wide distributed security system.