CERIAS Tech Report 2001-44

A Framework for Distributed Intrusion Detection using Interest-Driven Cooperative Agents

Rajeev Gopalakrishna
Center for Education and Research in
Information Assurance and Security
Purdue University, West Lafayette, IN 47907

A Framework for Distributed Intrusion Detection using Interest-Driven Cooperating Agents

Rajeev Gopalakrishna
Center for Education and Research in Information Assurance and Security,
1315 Recitation Building, Purdue University,
West Lafayette, IN 47907-1315
Email: `rgk@cs.purdue.edu`

Abstract

Current distributed intrusion detection systems are not completely distributed with respect to data analysis because of the presence of centralized data analysis components. This deficiency has many undesirable implications. Here we present a framework for doing distributed intrusion detection with no centralized analysis components. Our approach uses agents that are the only data analysis components. Agents cooperate by using a hierarchical communication framework. This cooperation is driven by interests expressed by the agents.

1 Introduction

During the last decade, research in intrusion detection has developed different approaches to doing intrusion detection. There has been a shift from a centralized and monolithic framework to a distributed one. But very little work has focussed on the nature of the communication mechanism between the components in a distributed intrusion detection system and the communication model in general. The data model and communication models have been chosen to suit the intrusion technique. In this paper, we look at the communication mechanisms in existing intrusion detection systems (IDS), current research efforts in this direction and identify the features of the communication mechanism for a distributed IDS. Based on the motivation derived, we propose a framework for distributed intrusion detection. We begin by looking at the general model of an IDS.

1.1 Model of an Intrusion Detection System

The Common Intrusion Detection Framework (CIDF) [24] defines a set of components that make up an intrusion detection system:

- **Event Generators (E-boxes).** The function of an E-box is to provide information about events to the rest of the intrusion detection system.
- **Event Analyzers (A-boxes).** Event Analyzers analyze the events obtained from the generators and look for any potential intrusive activity.
- **Event Databases (D-boxes).** The amount of data generated by the E-boxes and A-boxes is immense. Some of the data might need to be stored for post-mortem analysis. D-boxes define the means of storing those data.

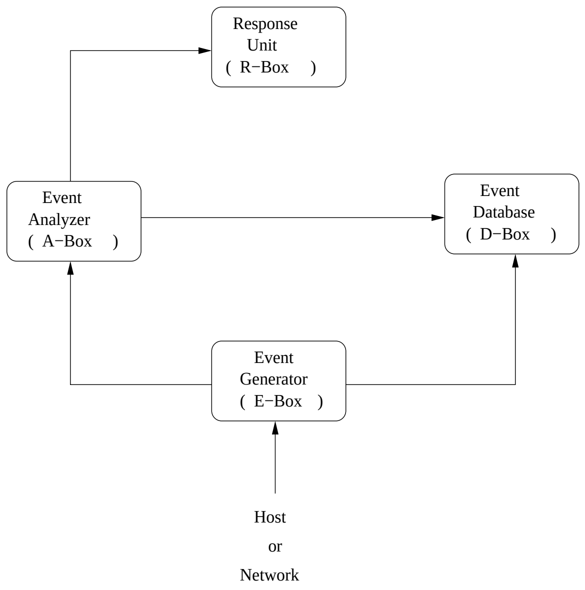

Figure 1: CIDF model of an intrusion detection system showing the interaction between the different components

- **Response Units (R-boxes).** Response units are components that enable the system to take countermeasures on detecting an intrusion by resetting connections, killing processes or altering file permissions.

These components together with the interaction as shown in Fig. 1 define a model of an intrusion detection system.

1.2 Distributed Intrusion Detection System

Spafford and Zamboni [28] define a distributed intrusion detection system as:

”a system where the analysis of the data is performed on a number of locations proportional to the number of hosts that are being monitored.”

We use the above definition in this paper. This definition suggests that merely having distributed data collection does not classify the intrusion detection system as being distributed. The analysis components should be proportional in number to the hosts being monitored, and should be distributed in their location. The data collection components and data analysis components are analogous to the E-boxes and A-boxes respectively.

Some of the features that distinguish different distributed intrusion detection systems are the following:

- Number and location of E-boxes
- Number and location of A-boxes
- Coordination between components
- Communication framework

We concentrate on the communication framework component of distributed intrusion detection systems. The framework consists of the communication model together with the actual communication mechanism. We discuss both of them in the following sections.

2 Communication Mechanism

Communication between the different components of a distributed intrusion detection system is a central part of the functionality of the system. It is by communicating messages that the components are able to obtain a global view of the system. Any disruption to the communication can cause the system to malfunction or possibly fail. In this section, we discuss the factors that determine the mechanism, the desired features and the present approaches.

2.1 Determining Factors

The factors discussed below are not mutually exclusive. Some depend on others. But they are discussed separately for clear understanding.

- **Number of components**. The number of components of an intrusion detection system depends on the expanse of the system being monitored. The system may be a single host, a local area network or an enterprise network spanning many independently administered domains. The communication overhead is an important factor with increasing number of components.

- **Location of components**. In a distributed intrusion detection system, the components are distributed across many hosts. There might be multiple components in a single host. The nature of the communication mechanism depends on the location of the components in consideration. Components within the same host might establish communication by means different from components in two different hosts. On this basis, the communication needs can be classified into two groups: intra-host communication and inter-host communication. Balasubramaniyan et al. [2] discuss in detail the choices available for each of them and their advantages and disadvantages. Recent research by Zamboni [32] is attempting to show that it is possible to perform intrusion detection using small sensors embedded in the code of the operating system. The communication is in this case different because the mechanisms discussed in [2] are for interactions between processes whereas sensors are not separate processes. So if the sensors have to communicate, they would have to use the kernel variables or data structures.

- **Type of data being considered**. There are different types of data involved in an intrusion detection system. Different components of the intrusion detection system might see different forms of data. The data may be raw audit trails, raw network traffic, reduced or compressed audits or alerts from other components. They may also be kernel variables and data structures, as may be the case in [32]. Having the same communication mechanism for all types of data though possible might not be optimal. For example, it would be infeasible to send all the log messages using a TCP connection considering the amount of log that is generated. Instead, it would be better to have a data analyzing component present at the host whose audit trail is to be monitored, and have the log read from the disk. The type of data influences the next three factors to a great extent.

- **Amount of data**. If the amount of data being considered is enormous as in the case of audit trails then it might be more efficient to place the analyzer components close to the collection components, such as in the same host. Thus, the amount of data can influence the location of components, which in turn affects the choice between intra-host and inter-host communication mechanisms.

- **Frequency of data generation**. If the frequency at which data is generated is high as in the case of observing a host audit trail or raw network traffic, it might be efficient to use a connection-oriented communication mechanism rather than a connectionless mechanism (with respect to inter-host communication). This is because, the frequent use of the communication mechanism would offset the overhead caused by the setting up of the connections. Whereas, in the case of alerts where the frequency of generation is comparatively less, it may be better to use a connectionless communication mechanism instead of using a connection that may end up being idle most of the time.

- **Method of data representation**. The method of data representation affects the size of the data, which in turn affects the choice of the communication mechanism. There have been efforts to standardize the representation of audit trails. For example, Bishop's standard audit trail format [5] defines a standard log record format that is both portable and extensible. Common Intrusion Detection Framework (CIDF) [16] and Intrusion Detection Working Group (IDWG) are some of the primary research groups that are making an effort to standardize data formats and protocols so that intrusion detection systems can share information and resources.

- **Sensitivity of data**. The components of an intrusion detection system have access to some of the most sensitive data about the hosts being monitored. Compromise of those data might be very harmful to the security of the system. Hence, the communication mechanism used to exchange those data between the components needs to be secure. The communication mechanism may need to ensure privacy and authentication. This has an impact on the performance of the system. However, all data might not be sensitive and that might offer a choice in the way they are exchanged. But again, it is difficult to decide what data are sensitive and what are not.

2.2 Desired Characteristics

When deciding to adopt a communication scheme for an intrusion detection system, it would be good to know the characteristics that we expect from the scheme. This way, we can compare the available schemes on these factors and choose the one that suits our needs. We briefly describe some of the features that might be desirable in a communication scheme for an IDS:

- **Reliability**
  A communication scheme is said to be reliable if it delivers error-free in-order data to the other end, even if there is a possibility of the data getting lost, corrupted, duplicated or being delivered out of order. A single missing message might cause the IDS to incorrectly identify an intrusion when none has occurred (false positive), or might cause the IDS to incorrectly fail to identify an intrusion when one has in fact occurred (false negative). This would make the IDS not only an unreliable component of the security system but also a factor leading to a false sense of security, which is even more dangerous.

- **Security**
  Given the increasing importance of an intrusion detection system as a prime component of the security system, it is reasonable to assume that the IDS itself may be a target for attack. An IDS can be attacked by compromising its components or by compromising the communication mechanism between the components. While compromising the components of an IDS would require taking over the host running the components, attacking the communication would require an access to the network that is comparatively easier in most situations. Such being the case, incorporating stringent security features into the communication mechanism seems logical. We discuss some security features that might be desirable in the communication mechanism and explain the rationale behind choosing them:

  - **Authentication**
    An attacker who is able to spoof messages can cause the IDS to fail noticing an attack or generate a false alarm. The IDS can potentially give away sensitive information about the monitored hosts if the messages are sent to unauthorized entities. Thus, it becomes important that the sender and the receiver of the message have confidence in the identity of each other. This necessitates a mutual authentication between the involved entities.
  - **Integrity**
    The sender of the message should have the confidence that the receiver will get the message unmodified. In other words, ensuring the integrity of messages exchanged between the components of the IDS is necessary to prevent malicious modifications of messages.
  - **Confidentiality**
    Messages exchanged between the IDS components can potentially contain extremely sensitive information such as passwords, host and network configuration details, the security state of the system etc, which might be of great value to an attacker. Therefore, maintaining the confidentiality of messages using cryptographic techniques becomes important. But cryptography comes at a cost of performance. So a compromise between confidentiality and performance should be made by trying to identify sensitive information that needs to be encrypted. However, the question of what is ”sensitive” is dependent on a number of factors, the prime one being the place of deployment of the IDS.
  - **Non-Repudiation**
    In a distributed IDS, the components of the IDS might be scattered throughout the network. Such being the case, it becomes important to be able to associate messages or actions with their originating entity. Though logging helps in keeping track of this, the communication protocol should necessarily provide a mechanism to tag messages with their origin in an infallible manner. Non-Repudiation depends on the authentication mechanism to know who someone is. So a weak authentication mechanism will be detrimental to the non-repudiating capability of the IDS.
  - **Non-Duplication**
    While the above factors may prevent an attacker from adding or modifying messages, the attacker can still can copy the message and send multiple copies to the receiver. This may confuse the receiver and may affect the accuracy of the IDS. Hence, detection of such duplicate messages is a desirable characteristic of the communication mechanism.
  - **Resistant To Denial of Service (DoS) attacks**
    An attacker can mount a denial of service attack on the IDS by making it difficult or impossible for the messages to be delivered. Even if the communication is not completely disrupted, because the IDS may depend on the time dependency of certain events (specially if the IDS is looking for anomalies), this sort of an attack may affect the synchronization and confuse or disarm the IDS. If the communication mechanism is resistant to such DoS attacks, the probability that the attacker will be thwarted increases.

  Ptacek and Newsham [26] document the different ways in which an IDS is susceptible to insertion and evasion attacks. This highlights the fact that an IDS should itself be resistant to attacks to be a reliable component of the security system. A survey of the research in intrusion detection systems by Axelsson [1] shows that very few intrusion detection systems [23, 25] address the security issues of IDS. AAFID [2, 28] discusses the security of an IDS to a considerable extent.

- **Scalability**
  The communication mechanism is one of the bottlenecks to the scalability of an IDS. Scalability

of the communication mechanism becomes important in the context of an IDS distributed over a large enterprise network. One can imagine hundreds of IDS components working in unison and cooperating by communicating among themselves. In such a scenario, the communication among the IDS components should not add significantly to the overhead at the host and network, and should minimally affect their performance.

- **Speed**
  For an IDS to operate in real time (or near-real time), fast transfer of messages from one part of the system to another becomes necessary. This not only requires the underlying communication mechanism to provide good transmission times but also requires the use of fast and efficient algorithms at both ends to meet the above needs.

2.3 Present Approaches

In this section, we look at the different communication mechanisms used in intrusion detection systems. We also look at the current efforts in this direction. We briefly outline the features of each mechanism and discuss its suitability in intrusion detection systems.

- **Transmission Control Protocol (TCP)**
  TCP is a reliable, connection-oriented, stream protocol in the Internet Protocol suite. It guarantees error-free, in-order delivery of data. It has a number of features such as congestion control, slow start etc. that make it favorable for use over large networks with varying latency, bandwidth and congestion characteristics. Most intrusion detection systems use TCP as the underlying transport protocol for the above reasons. But there are certain features of TCP that we can argue as being unfavorable for intrusion detection systems. They are as follows:

  – TCP causes the flows to respond to the congestion state of the network. This prevents flows from being over-aggressive and promotes fairness among the different flows. Some questions arise: Can a TCP connection in an IDS be treated on par with a TCP connection carrying HTTP traffic from websites? Will it be appropriate for TCP connections in an IDS to slow down and perform congestion control in the wake of a DoS attack, when you actually want it to be more alert and faster than ever?
  – The overhead for providing a reliable and connection-oriented communication might affect the scalability and performance of an IDS.
  – TCP lacks features such as encryption that would be desirable in an IDS.
  – TCP is susceptible to attacks such as sequence number attacks and SYN flooding to name a few.

  Reliable transfer of messages is an important criterion for an IDS. The use of UDP might not be favorable because it is an unreliable protocol. But it offers less overhead because it is a connectionless protocol. Moreover, it does not do congestion control. So a reliable protocol built over UDP or reliable UDP (RUDP) [22] might seem to be a better option. There is research being done in this direction that we discuss later in this paper.

- **SSH**
  SSH (Secure Shell) is a protocol for secure remote logins and other secure services over an insecure network. Its basic use is to serve as a secure alternative to telnet and rsh. But it also allows one to transparently and securely forward TCP connections from one machine to another. This feature can be used to achieve encrypted communication for arbitrary TCP connections. But this requires a

SSH connection between the machines involved. And because it is built on top of TCP, it shares its problems. AAFID [2] uses a SSH connection for executing a starter program in a remote host that facilitates the activation of new entities in that host.

- **Simple Network Management Protocol (SNMP)**
  The Simple Network Management Protocol (SNMP) [29] is used to refer to a collection of specifications related to network management that include the protocol, the definition of data structures, and associated concepts. The SNMP model of network management includes the following key elements:
  - Management Station (SNMP Manager)
    The SNMP Manager contains applications for analyzing the information obtained from the agents, an interface to allow one to monitor and control the network, and a database of information obtained from the MIBs of all the monitored entities in the network.
  - Management Agent (SNMP Agent)
    All the devices in the network that need to be managed have a SNMP Agent running in them, which can be controlled from the manager. The agents respond to requests for information and actions from the manager. They can also send information to the manager asynchronously.
  - Management Information Base (MIB)
    There are different aspects of every device that can be controlled. Each such aspect can be represented by a data variable. A database containing all such variables for a device is called a MIB for that device. The SNMP agent running on the device has access to the MIB of that device. So the MIB serves as a collection of access points at the agent for the manager to manage the device. A manager can monitor the state of a device by retrieving the values from the MIB. It can initiate some action at the device by setting values in its MIB.
  - Network Management Protocol (SNMP)
    SNMP is a network management protocol used for the management of TCP/IP networks. It is an application-level protocol that operates over UDP. Because of this, SNMP is also connectionless. The manager and the agents communicate using SNMP, which offers three key primitives: `get` allows the manager to retrieve values of objects in the MIB at the agent, `set` allows the manager to set the values of objects in the MIB at the agent, and `trap` allows agents to notify events to the manager asynchronously.

  The Remote Network Monitoring (RMON) specification is another component of the SNMP standards. It supports a remote monitoring MIB that provides the manager with vital information about the monitored network. This information can include traffic statistics, alarms, host statistics, captured packets among other things. So a monitor with RMON capability is used to monitor the network as a whole instead of individual devices.

  SNMP is the industrial *de facto* standard for network management. Intrusion Detection is fast becoming a necessary component of any network management framework. So an IDS based on the SNMP model can easily be integrated into such a framework. Moreover, the SNMP model has a lot of features that can be used in an IDS. The concept of manager and agents supported by SNMP is an inherent feature of most intrusion detection systems. The SNMP model not only supports passive monitoring but also allows a manager to issue a command to an agent to perform an action. This is done using the SNMP `set` primitive, in a process called *action invocation* - an object in a MIB can be used to represent a command so that a specific action is performed if that object is set to a specific value. The agents can raise alarms asynchronously using the SNMP `trap` primitive. The RMON functionality can serve as a foundation for a network-based IDS. The earlier versions of SNMP (v1 and v2) lacked security features. But SNMPv3 has provisions for privacy and authentication of messages.

JiNao [14] is an intrusion detection system for network infrastructure that uses SNMP as the underlying model. Each router/switch is associated with a local intrusion subsystem. These subsystems interact by accessing MIBs of each other using remote MIB agents. This work also looks into other details of SNMP that can be used in an intrusion detection framework. AAFID [2] suggests SNMP as an alternative framework for implementing the AAFID architecture. Some commercial intrusion detection systems use SNMP traps as a means of notifying alerts to a management console such as HP OpenView, Tivoli Netview or Cabletron Spectrum.

Contrary to its name, (i.e. ”Simple” Network Management Protocol), SNMP is a highly complicated protocol to implement. With simplicity being the key to security, adopting the SNMP model for an IDS may not be a good idea. However, there have been suggestions to adopt the naming convention used in SNMP. Bass [4] suggests that the SNMP MIB model might be well suited for defining a TCP/IP threat taxonomy and provides examples of security threats represented using the ASN.1 MIB notation (ASN.1 refers to ”Abstract Syntax Notation One”, which is an obscure type declaration language, adopted by SNMP to represent the MIB entries). Structure of Management Information (SMI), which identifies the datatypes that can be used in the MIB and specifies how objects in the MIB can be represented and named, was one of the choices sought after by the IETF working group on intrusion detection (IDWG) for the data format for messages exchanged in intrusion detection systems. They have released an internet draft [17] that compares SMI and XML (Extensible Markup Language) and details the rationale for choosing XML over SMI. Inspite of these drawbacks, we feel that more research is needed to see how the field of intrusion detection can apply some of the features of SNMP.

- **CIDF**
  The Common Intrusion Detection Framework (CIDF) [16] is an effort to develop protocols and interfaces for intrusion detection systems to cooperate. We concentrate on the communication mechanism of CIDF [15]. The CIDF message layer protocol provides support for reliability, privacy, authentication and data integrity among other things. It suggests reliable CIDF messaging over UDP as the default transport mechanism and proposes a negotiation mechanism to choose other variations of the transport mechanism. So the transport mechanism could be UDP, reliable UDP or TCP. The transport negotiation mechanism is a good measure that helps accommodating both simple and complex devices with varying requirements for security services into the IDS framework.

- **IDWG**
  The Intrusion Detection Working Group (IDWG) is an IETF working group that had its origin in the CIDF group. It too aims at defining data formats and exchange protocols for sharing of information among intrusion detection systems. The difference seems to be that IDWG is more oriented towards the vendors of commercial intrusion detection systems in its approach.

  They propose an Intrusion Alert Protocol (IAP) [10], which is an application-level protocol for exchanging intrusion alerts among intrusion detection components. IAP uses TCP as its transport protocol for reliable delivery of data. It also makes it mandatory to use the Transport Layer Security (TLS) 1.0 protocol [7] to negotiate the security parameters. Along the same line, verification of certificates for the entities involved in the protocol is a must, which ensures mutual authentication. Gupta et al. [10] discuss in detail about the various features of the IAP protocol. The IDWG is also working on the data format and other protocols.

The present approaches suggest that identifying a single communication mechanism for an IDS in the presence of devices of varying complexities is a difficult task. As of now, using a simple default mechanism to negotiate other complex mechanisms seems to be ideal.

3 Distributed Communication Models

A distributed system consists of several computing entities connected to a network that serves as the communication channel. As current intrusion detection systems tend to be distributed in nature, it might be relevant to look at the distributed models of communication.

A distributed communication model indicates the pattern of communication among the participating entities. It answers questions such as: what are the roles of the participating entities? which entities can request information? how does an entity know the availability of information? how is information disseminated to the interested entities? etc. The model is independent of the underlying communication mechanism that is actually used to transfer the information. The communication model determines the level of coupling between the entities, which in turn contributes to the scalability of the system as a whole. Hauswirth and Jazayeri [11] discuss in detail and compare the distributed communication models. There are essentially two kinds of models: the **client-server model** and the **peer-to-peer model**. While the client-server model deals with two participating entities, the peer-to-peer model deals with many peer entities. And the entities in the peer-to-peer model are referred to as producers and consumers as opposed to clients and servers . A distributed IDS consists of multiple entities producing and consuming information about the state of the system. For this reason, a peer-to-peer model is more relevant in this context. The peer-to-peer model may again be divided into two subclasses namely:

- **event-based model**
  In this model, any entity may produce events and any entity may consume events. Thus, the entities are loosely-connected and their roles are symmetric. This leads to higher scalability as there is very little coupling. Entities advertise the events they produce. Entities also declare interest in receiving specific events from other entities and are notified on the occurrence of those events. This requires an expressive advertisement and subscription language.

- **push-based model**
  In this model, certain entities are designated as producers and others as consumers. Logical channels connect producers with interested consumers. This leads to tighter coupling and makes the model less scalable.

As scalability is a key issue in distributed intrusion detection systems, their framework should be based on an event-based communication model. This might make their design complex as issues related to event advertisement, event subscription and event notification have to be handled.

4 Intrusion Detection vs Intrusion Prevention

Real time intrusion detection attempts to detect if the system is being attacked while the attack is being mounted. A far greater challenge is to detect such an attack and react in such a way so as to prevent it from achieving its objective - *intrusion prevention*. Due to the underlying design, intrusion detection systems work in parallel with the execution of the intrusion and often lag behind. For example, a host-based IDS that relies on post-event log analysis can only identify suspicious events/actions and a network-based IDS still allows a packet containing malicious code to reach the destination during the time it takes to copy the packet and detect that it is malicious in nature.

Fig. 2 gives a logical representation of an attack. The *attack object* could be a network packet containing malicious code or it could be a piece of malicious code in a host. The target could be any (hardware or software) component of a host. Let t_a be the time it takes for the attack object to mount the attack on the target. If the intrusion has to be prevented, then we need to detect that the object is hostile in nature,

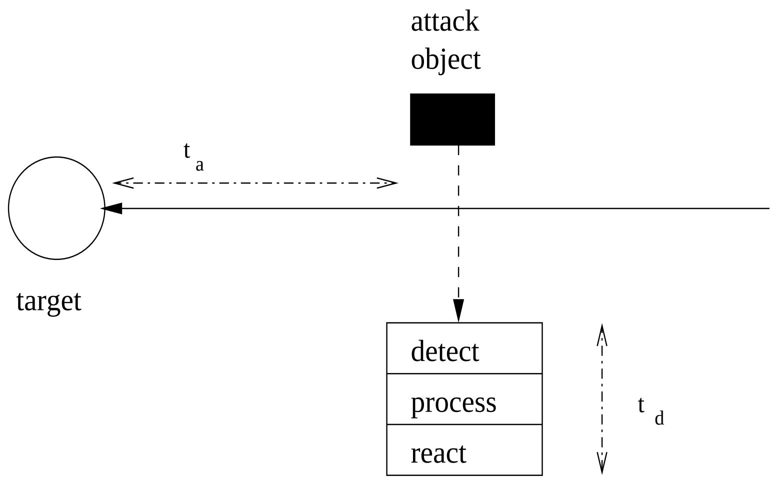

Figure 2: A representation of an attack showing the attack object and the target. t_a is the time it takes the attack object to mount the attack on the target. t_d is the time it takes to prevent the attack from happening

determine the nature of the attack it is trying to mount and the target it is aiming at, and react to the attack in such a way so as to prevent it. Let t_d be the time required to do the above. If the intrusion has to be successfully prevented then t_d should be less than t_a. Bass [4] defines the ratio of t_a to t_d as the *Gamma Factor*. The paper also suggests that it might not be mathematically feasible to do intrusion prevention using in-band communication. In other words, an intrusion detection and response system cannot be successful if it uses the same communication links as the traffic it is monitoring. Bass [4] suggests that the components of such a system should be on a network that is out-of-band and faster than the monitored network.

5 A Futuristic View

In the future, the software and the hardware might be intrinsically more secure than it is now. But this will not eliminate the presence of new vulnerabilities and new attack strategies. So intrusion detection will still be necessary. The IDS components might become more tightly coupled with the system as such. The work by Zamboni [32] takes a step in that direction by attempting to embed ”sensors” in the operating system code.

Estrin et al. [8] envision that unattended autonomous sensors that coordinate amongst themselves to achieve a large sensing task will revolutionize information gathering and processing. They propose that *localized algorithms* wherein simple local processing achieves a desired global objective, maybe necessary for scalable sensor coordination. They also discuss *directed diffusion*, a simple communication model needed for describing localized algorithms. In this model, sensors express and propagate interests and interests establish gradients that direct the diffusion of data. As intrusion detection is essentially gathering and processing of information looking for malicious code/actions, the above work is relevant to this field.

6 Motivation

Over the last ten years, research in the field of intrusion detection has been heading towards a distributed framework of monitors that do local detection and provide information to perform global detection of intrusions. A few of the intrusion detection systems that adopt this methodology are DIDS [27], GrIDS [30], EMERALD [25] and AAFID [2, 28]. Spafford and Zamboni [28] define such systems as distributed intrusion detection systems based on the location and number of the data analysis components. All these systems are hierarchical in nature. The local intrusion detection components look for local intrusions and pass their analysis results to the upper levels of the hierarchy. The components at the upper levels analyze the refined data from multiple lower level components and seek to establish a global view of the system state. We argue that such a distributed intrusion detection system is not completely distributed with respect to data analysis because of the centralized data analysis performed at the higher levels of the hierarchy.

The motivation for research is derived from three sources: the concept of agents to perform intrusion detection [6, 28], the event-based communication model [11] and the concept of interest propagation [8]. We propose an architecture for employing distributed agents that are autonomous but cooperate to perform intrusion detection in a distributed fashion. Our architecture is unique in that it is based on agents cooperating using an **interest-based** communication model and that all the data analysis is done by agents without the presence of any **analysis hierarchy**.

Under certain assumptions, distributed intrusion detection can be performed without the need for communication between local detection components as shown by Hofmeyr [12]. But under general conditions, a completely distributed analysis requires global correlation and intelligent coordination among the distributed analysis units, which can introduce a significant resource overhead as noted in EMERALD. In our current research effort, we propose a framework for a distributed intrusion detection system and expect that by employing agents and an interest-based communication scheme, we will minimize the resource overhead caused by a completely distributed analysis.

7 Previous Work

DIDS [27] is a distributed intrusion detection system consisting of host managers and LAN managers doing distributed data monitoring and sending notable events to the director. The centralized director then analyzes these events to determine the security state of the system as a whole. The centralized director is clearly the bottleneck to the distributive approach of DIDS. And as there is only one level of hierarchy with all host and LAN managers reporting to a single director, it lacks scalability.

EMERALD [25] is a framework for performing distributed intrusion detection. It employs monitors at the levels of hosts, domains and enterprises to form an analysis hierarchy. It uses a subscription-based communication scheme both within and between monitors. But the inter-monitor subscription scheme is hierarchical thus limiting access to the events or results from the layer immediately below.

GrIDS [30] constructs activity graphs representing hosts and network activity. It models an organization as a hierarchy of departments and hosts. Activity that crosses departmental boundaries is passed up to higher levels of the hierarchy. It uses an aggregation approach to infer and reduce the data that must be analyzed at the higher levels. AAFID [2, 28] is a framework for a distributed intrusion detection system that employs autonomous agents at the lowest level for data collection and analysis and transceivers and monitors at the higher levels of the hierarchy for controlling the agents and obtaining a global view of activities. It suggests the use of filters within hosts that are data selection and abstraction layers, providing a subscription-based service to agents.

Such hierarchical distributed intrusion detection systems have the following drawbacks:

1. **Analysis hierarchy**
There is a hierarchy in the data analysis. Data analysis takes place at all levels of the hierarchy. This means that in the wake of a new distributed attack, changes may have to be made in modules at many levels.

2. **Data Refinement**
Data refinement takes place across levels with each level only reporting the notable events to the higher level. We argue that the knowledge of what events are important on a system-wide level is circumstantial and is difficult to infer at the lower levels of the hierarchy. If the refinement is strict, we may end up losing some system-wide notable events and if the refinement is loose, the higher level analysis modules will be flooded with large amounts of data from the lower levels. Finding an a priori suitable compromise may be difficult.

3. **Bulky modules at all levels of hierarchy**
Analysis engines based on either signature recognition or anomaly detection are often large modules analyzing system audit logs, user activities and system state. This consumes a significant amount of resources in terms of CPU usage, disk I/O and memory usage. In the systems mentioned above, these components are present at all levels of the hierarchy. Such components present multiple points of failure. Degrading or disabling a top-level component would severely limit the detection capability of the system. Moreover, such bulky components are expensive to replicate to achieve fault tolerance.

4. **Passive Interaction**
The components of the intrusion detection system interact with each other in a passive way. The lower level components generate data for the upper level components as per the rules driving them. There is no mechanism for a component to query other components on the basis of some analysis that it has done. The subscription-based communication mechanism between monitors in EMERALD and between the agents and filters in AAFID offers a mechanism for active interaction. But in EMERALD, it is limited to occur between the adjacent levels of the hierarchy and in AAFID, it is allowed only within a host. Ning et al. [20, 21] recognize the importance of the querying facility in cooperative intrusion detection systems. They propose an extension to the common intrusion specification language (CISL) [9] to allow intrusion detection components to specify requests for particular information from other components.

In the paper by Crosbie and Spafford [6], which is one of the first works proposing the use of autonomous agents for intrusion detection, the authors propose broadcasting suspicions of intrusion among agents as a means of cooperating to achieve the common goal of intrusion detection. Broadcasting data is highly infeasible in a large network with hundreds of agents. In the works of Barrus and Rowe [3] and Ingram [13], there appears to be only one agent per host and the agents on different hosts transmit all alerts to all other agents that they know using TCP connections. This will not scale well because of the communication overhead. Mell and McLarnon [18] attribute this problem with hierarchical intrusion detection systems to the location of components being static and propose modeling these components as mobile agents. But mobile code comes with its own problems namely security concerns and restricted execution environments. CARDS [31] adopts an approach of generating and distributing ”detection tasks” among monitors to cooperatively detect attacks. Detection tasks are parts of an attack signature, which the authors also refer to as ”predefined queries” [21]. So, queries are implicitly defined in the signature pattern of an attack and there appears to be no support for active querying in the architecture.

Our approach at solving this problem differs in the sense that it involves intelligent coordination among agents that are the only analysis components communicating actively using a hierarchical framework.

8 Our Approach

In our approach, the key effort is directed towards making the agents cooperate in an intelligent manner by allowing them to communicate actively with each other. The analysis hierarchy is avoided by performing all the analysis only at the agent level. The intelligence in cooperation is attempted by communicating events

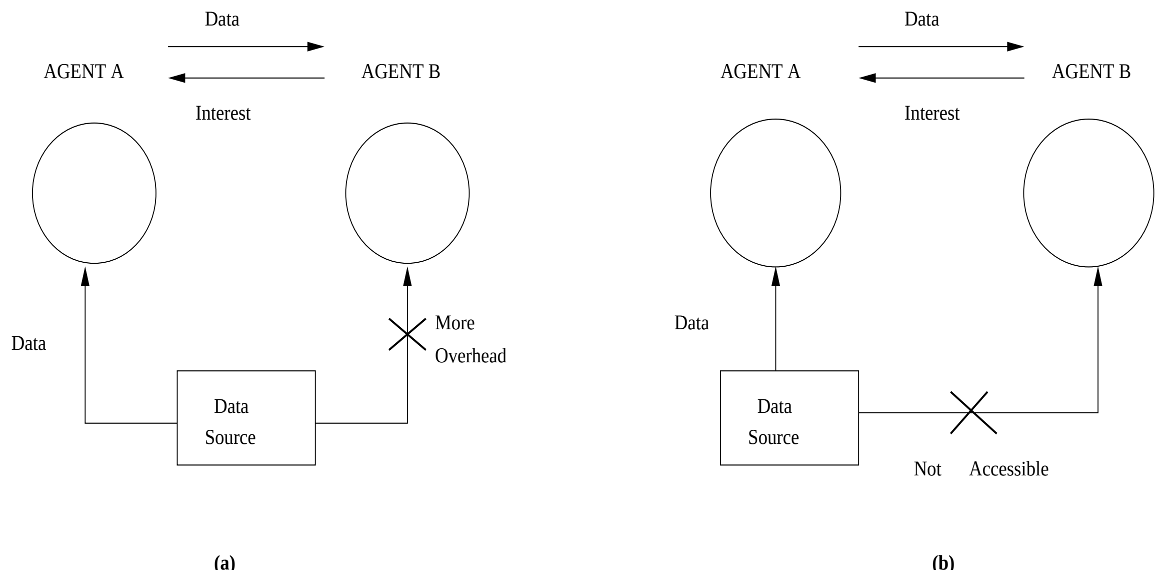

Figure 3: Generation of an interest based on locality of data collection. (a) To minimize overhead. Agent B can collect the data itself, but the overhead of obtaining it is larger than requesting it from Agent A. (b) Inaccessibility of data. Agent B does not have access to the data it needs, so it requests it from Agent A which has access to the data.

or alerts to only those agents that are interested in them. Agents propagate their interests in the network in a hierarchical fashion. The mechanism that enables this is handled by lightweight modules at the upper levels of hierarchy. Agents can specify new interests in response to notification of some events or alerts which is why we use the term ”active” with respect to communication. The following sections elaborate on agents, interests and the components of our architecture.

9 Agents

The definition and advantages of autonomous agents are discussed in detail in AAFID [2] and the paper by Crosbie and Spafford [6]. We retain all the features of agents proposed in AAFID. In addition, the agents in our approach are cooperative and dynamic. By cooperative, we mean that the agents respond to requests for events or alerts from other agents. And based on the system state, agents can request different events and alerts from the other agents. This adds dynamism to the behavior of the agents and allows them to assess the system state in a better way than simply analyzing those events and alerts reported to them as determined at their initiation. This also helps reduce the overhead resulting from agents observing all events at all times and reporting them to a higher authority [2, 28].

Each agent performs a certain predefined security monitoring function at a host. The function of an agent may pertain to the security of the host or the network to which the host is attached. Agents performing the former function may be necessary at every host. Agents guarding the network may be distributed over the hosts in the network or may even be duplicated across hosts for accuracy and fault tolerance. So certain agents may be present only in specific hosts in the network.

10 Interest

We define an **Interest** as

”a specification of data that an agent is interested in, but is not available to the agent because of the locality of data collection or because the agent was not primarily intended to observe those data”.

Some situations in which an agent will express an interest because of the locality of data collection are as follows:

- On a single host or network, there may be more than one agent that needs data from the same data source. Having each agent obtain the data on its own leads to duplication of effort in terms of accessing the data and parsing it to obtain the useful information. So in cases where the overhead of the data access mechanism is more than the communication overhead of transferring the data between agents, a more efficient approach will be to have one agent obtain the data and make it available to the other agents as shown in Fig. 3(a). An example of such a situation is agents in the same host wanting to access the information from log files. Having every agent read the log files, parse them and obtain the needed information may lead to more overhead in terms of CPU usage, disk I/O and memory usage than having one agent do all this and send the information to the other agents using some sort of inter-process communication.
- In a situation where an agent is intended to detect coordinated or distributed attacks, the agent may need data from agents present in multiple hosts in the network. This is different from the previous situation because in the previous case, all the agents could access the data source. The choice of having one agent access the data source and send the data to the other agents was made because it offered less resource overhead. Whereas, in this situation, an agent may need data from other agents because it does not have access to the source of the data as shown in Fig. 3(b).

An agent may also need some data only under certain conditions. These conditions can arise on being notified of an event or an alert. So in such situations, the agent will express interests for those data because of its conditional requirement. In these cases, it will be a waste of resources to unnecessarily supply the agent with those data all the time. This is a major drawback of existing hierarchical intrusion detection systems. Because of the lack of any active communication, the lower levels end up sending all the data at all times to the upper levels. Seeking data refinement as a solution might result in loss of some notable events.

10.1 Interest Propagation

In a large enterprise network, there may be thousands of hosts connected to different, independently administered domains. In this scenario, an agent cannot keep track of all the other agents in the network. So while an agent may know what data it is interested in, it may or may not know if there are any agents collecting that data and if so, it may not know the location of the agents that are collecting that data. Hence the propagation of interest in the network becomes important. To a certain extent, we derive inspiration for interest propagation from the work of Estrin et al. [8]. We propose a hierarchical propagation of interests. This hierarchy is divided into three levels, namely local, domain and enterprise. We borrow these terms from EMERALD. While we argue against the use of analysis hierarchy, we use a hierarchical framework to propagate interests. This functionality can be added to the different levels of the hierarchy without the use of ”bulky” modules. Moreover, this functionality need not be altered when agents are added or removed.

10.2 Types of interests

An agent may have some knowledge of its interest in certain data. This information is included in the interest and it directs the entities processing the interest to handle it in a certain fashion. On this basis, we classify interests into different types:

10.2.1 Directed or Propagated Interests

An agent may or may not have the knowledge of the host or domain from which it is interested in getting the data. On this basis, we can classify the interests as:

Directed Interest
In certain situations, an agent may know the host or the domain from which it is interested in getting the data. In such cases, the interest will be directed only to that host or domain. Such an interest is termed as a directed interest. If there is an agent in the targeted area that collects the data mentioned in the request, then it will service that interest.

Propagated Interest
If the agent has no specific host or domain it is interested in, then the interest is propagated across the whole enterprise. Any agent that can service that interest will do so. We term such an interest as a propagated interest.

10.2.2 Local, Domain or Enterprise level Interests

Based on the level of the hierarchy to which an agent restricts the propagation of its interest, we identify three types of interests namely:

Local level Interest
If an agent is interested in getting data only at the local host, then such an interest is termed as a local level interest. This may happen when only certain agents in the host are assigned to collect certain data to avoid duplication of effort.

Domain level Interest
If an agent is interested in getting data only in the local domain, then such an interest is termed as a domain level interest. This may happen, for example, when agents are attempting to detect coordinated or distributed attacks in their domain or when they are authorized to obtain data only in their domain.

Enterprise level Interest
If an agent is interested in getting the data from anywhere in the enterprise, then the expressed interest is termed as a enterprise level interest. This may happen when agents are attempting to detect an enterprise-wide attack. This is the same as a propagated interest, unless the framework is employed across multiple enterprises. In this case, a propagated interest will be propagated across multiple enterprises while an enterprise level interest is restricted to the enterprise in which it originated.

10.2.3 Permanent or Temporal Interests

Interests may also be classified based on the duration for which the agent needs the data specified in the interest. They are the following:

Permanent Interest
Interests specified by agents for the duration of their existence are termed as permanent interests. Agents might specify permanent interests under the following example situations:

- An agent may not able to collect the data it is interested in because the data is generated elsewhere and there is no way of collecting it at this agent (as shown in Fig. 3(b)).
- The data is being collected by another agent and it is less expensive in terms of CPU usage, disk I/O and memory usage to transfer the data between the agents all the time than to replicate the data collection mechanism at this agent (as shown in Fig. 3(a)).

Temporal Interest
Temporal Interests are the interests specified by agents for a certain duration of time. Such interests may be expressed by agents in response to certain events or alerts. The agent(s) servicing the interest continue to do so only for the duration of time as specified in the interest. This feature allows agents to respond to changes in the system behavior. It avoids the need to monitor all the data at all times by transferring data only on demand.

10.3 Granularity of interests

The data that we have been referring to can be either an event or an alert. An **event** is any data that has not been analyzed. An **alert** is the result of analyzing a set of events that indicates some potential intrusive activity. An agent can express an interest for an event or for an alert. This flexibility allows the agents to control the intensity of their monitoring. For example, an agent can subscribe to alerts from other agents initially and on observing certain system behavior, it can express interests for the events constituting the alerts to perform a detailed analysis. In this way, an agent can increase its ”curiosity level”. This adds dynamism to the agents and reduces the overhead caused by continuous monitoring of events.

11 Data Delivery

The data that is requested by an agent in an interest is delivered by the agent(s) servicing that interest. Currently, we see two possible ways of doing data delivery:

- Data delivery can be done using the same hierarchical framework that is used to propagate interests (see Fig. 4(a)).
- Data can be delivered to the agent directly without going through the hierarchy (see Fig. 4(b)).

Both have their advantages and disadvantages. We compare them below:

- **Failure of modules**
  In the event of modules in the higher levels of the hierarchy failing or being degraded or disabled by an intruder, there is a significant impact on the detection capability of the intrusion detection system as a whole. If data transfer between agents takes the path through the hierarchy then it may be stifled because of the dysfunctional module(s). On the other hand, if we transfer data directly between agents then the dysfunctional modules will only affect future interest propagation. The current detection activities will proceed normally as interests have already been registered and the flow of data is unaffected. This is true assuming that the affected module is not physically mapped to a router that is in the path of the connection between the agents.

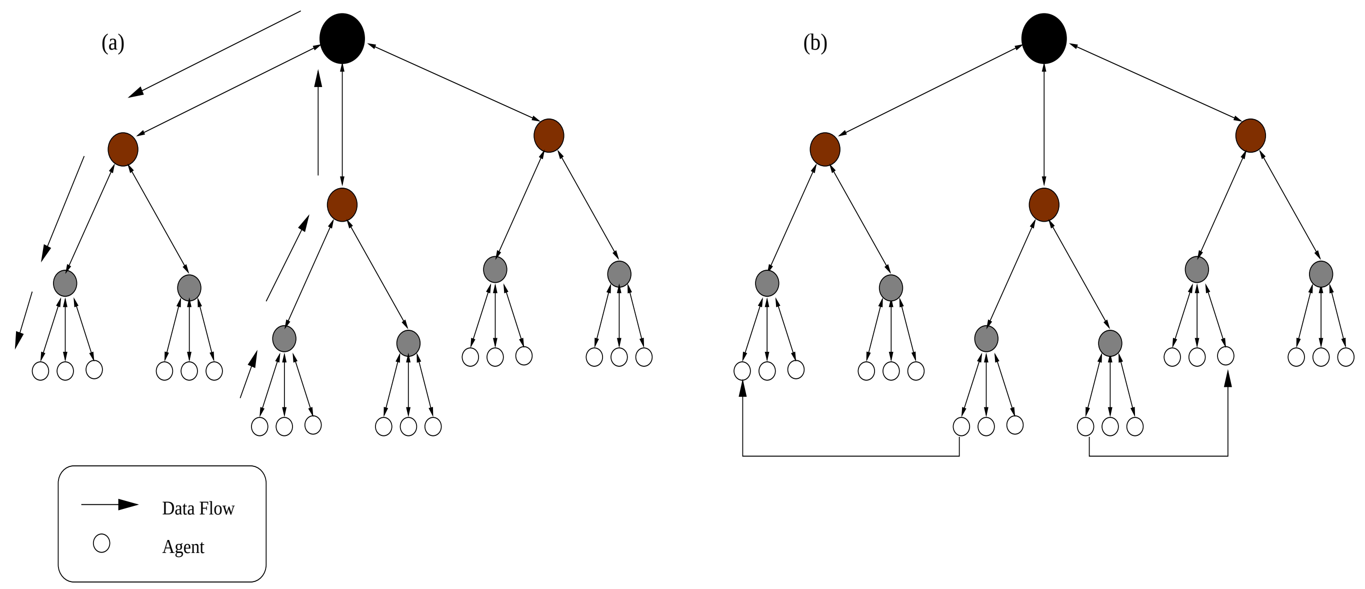

Figure 4: Delivery of data to the agent expressing an interest (a) Using the path through the hierarchy (b) Using a direct connection

- **Scalability**
  In a large enterprise, there may be thousands of hosts and a proportional number of agents. In such a scenario, having agents communicate data directly with each other may lead to thousands of connections in the worst case. Even if we consider optimizations such as agents sharing a connection when communicating with agents on the same destination host, the number might be enormous if we have hundreds of agents on different hosts servicing the interest of a single agent. This lack of scalability with respect to the number of connections and the load on the hosts will not be an issue if we use the hierarchical framework for data delivery.

- **Data Coalescing**
  If an agent is servicing the same interest to multiple agents then optimizations can be done by sending a single copy of the data through the common path in the hierarchy to the destination agents, duplicating it at appropriate levels of the hierarchy and dispatching it to the individual agents. This type of data coalescing is possible only if the serviced agents belong to the same host in the case of direct communication of data between agents.

Having looked at the advantages and disadvantages of the two possibilities, we could also think of incorporating both the mechanisms in the framework. There can be situations where one is better than the other. For example, in the case of an agent expressing a permanent interest in data from a non-local agent, it may be better to have a direct connection between the two hosts containing the agents. But in the case of agents specifying temporal interests, using the existing hierarchy for data transfer may be better than establishing a new connection between the two hosts of the agents and breaking it after the time period is over.

We propose to explore these possibilities in greater detail. But for the present discussion, we assume the use of the hierarchical framework for both interest and data transfers.

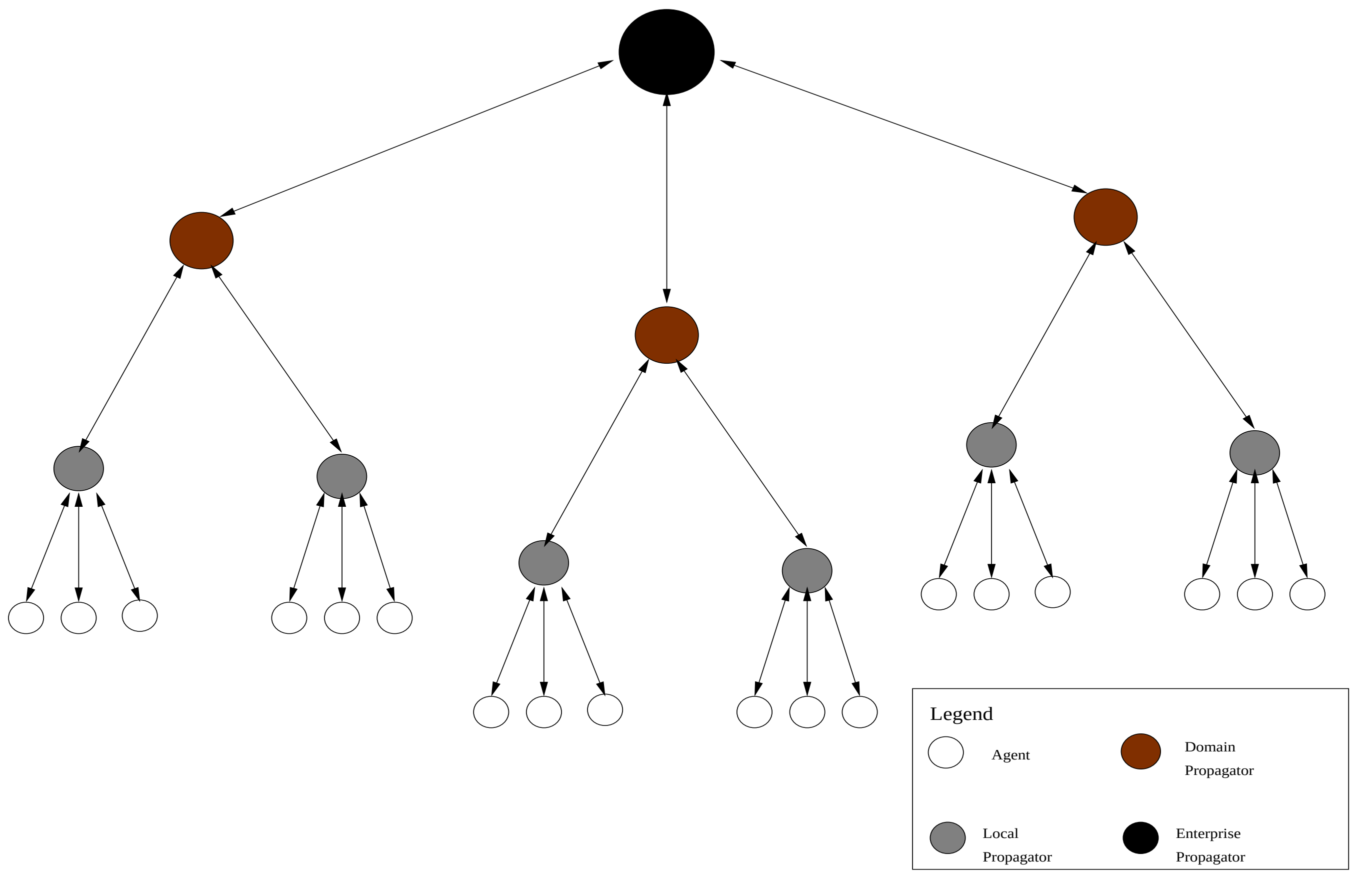

Figure 5: Logical organization of agents and propagators showing the communication hierarchy. The bidirectional arrows represent the flow of both interests and data between the entities.

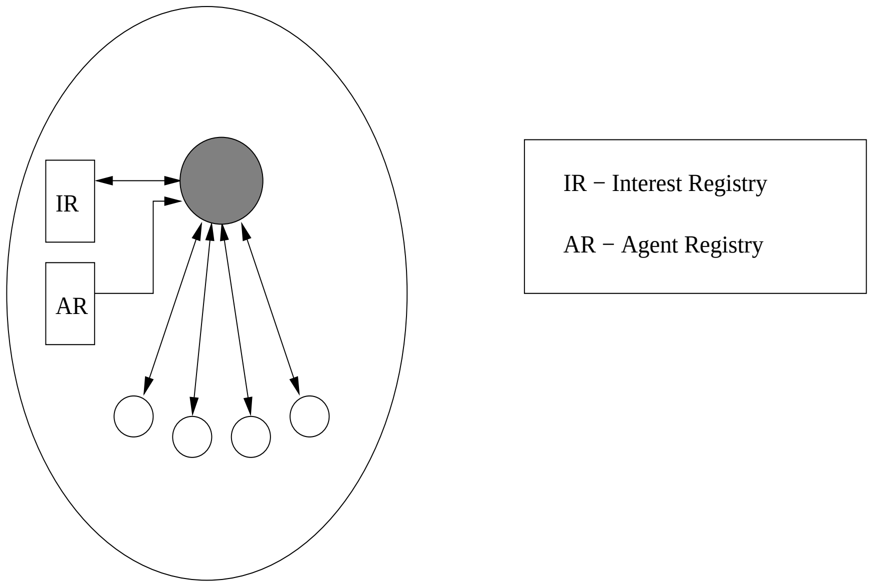

Figure 6: Physical layout of the components in a host showing agents, interest and agent registries and the local propagator

12 Component Description

In this section, we describe the components of the architecture as shown in figures 5 and 6:

12.1 Agent Registry

The agent registry is present on all hosts running agents. It maintains information about the agents running in the host, the events they collect and the alerts they generate. This information is used to determine if there are any agents on the host that can service an interest.

12.2 Interest Registry

The interest registry is used to keep track of both interests originating at the agents in the host and interests being serviced by the agents in the host.

12.3 Propagators

Propagators are modules that transfer interests and data between the different levels of the hierarchy. Propagators at one level are connected to all the entities (propagators or agents) in the level below them and to exactly one other propagator in the next higher level. They need to maintain information about the entities in the level below them so that they can forward the information (interests or data) to the right entity. They need not maintain any state information about the interests and data transferred between the agents. Thus, the propagators are relatively very lightweight entities when compared to the modules at the different levels of hierarchy in the systems mentioned in section 7. Moreover, only their information database needs to be updated when any entity is added or removed in the level below them. Their functionality is unaltered. This is in contrast to the systems described in section 7 where every time a new agent is added or removed, the functionality of modules in the path from the agent to the top of the hierarchy needs to be updated for them to modify their response to the events or alerts sent by this agent. This makes the installation and removal of agents easier.

Based on their level in the hierarchy, propagators can be of three types:

12.3.1 Local Propagator

Local propagators are located in every host. They provide a link between the agents and the domain propagator for moving interests and data back and forth.

On receiving an interest from a local agent, the propagator determines the type of interest. Unless it is a host or domain directed interest to which it does not belong, it refers to the agent registry and determines if there are any agents on this host that can service the interest. If so, it informs those agents to send the requested events or alerts to it and also updates the interest registry. If there are no agents on the host that can service the interest and it was a local level interest, it informs the requesting agent about the inability to service that interest. In all other cases, it also forwards the interest to the domain propagator.

On receiving an interest from the domain propagator, it looks up the agent registry to determine if there are any agents on this host that can service this interest. If so, it informs those agents to send the requested data to it and updates the interest registry. If not, it discards the request. The local propagator could also send an ACK to the requesting agent if there was any agent on its host that could service the interest. This way, the requesting agent would know if its interest will be serviced at all. We propose to test the applicability and feasibility of this mechanism in future.

On receiving data from a local agent, it looks up the interest registry and sends the data to all the local agents whose interest matches the data received. If there are any foreign agents that have registered an interest for the same data, it forwards the data to the domain propagator which handles the further transfer.

On receiving data from the domain propagator, it looks up the interest registry and sends the data to all the local agents whose interest matches the data received.

If any temporal interests have been registered by foreign agents at the host, then the local propagator forwards the data only for that duration of time. At the end of that duration, the local propagator informs the local agents servicing that interest to stop sending the data unless there is some other similar interest that still needs to be serviced. It then removes the expired interest from the registry.

The functionality of domain and enterprise level propagators is much simpler when compared to this.

12.3.2 Domain Propagator

A Domain propagator is present in every domain. It has knowledge of all the hosts in the domain and is connected to all of them. It is also connected to the enterprise propagator. The function of the domain propagator is to propagate interests and data among hosts, both within the same domain as well as in separate domains. Note that the domain being referred to, is a security domain.

An interest received from the enterprise propagator is handled based on the nature of the interest. A host-directed interest is forwarded only to that host. A domain-directed or a propagated interest is sent to all the hosts to which it is connected.

An interest received from a host in its domain is again handled based on its nature. An interest directed to a host in the same domain is forwarded to that host. If the directed host is not in its domain, it forwards it to the enterprise propagator. Similarly, a domain directed interest is sent to all hosts in the domain if the domain specified is its own domain, else it is forwarded to the enterprise propagator. Domain level interests are forwarded to all hosts in its domain. Enterprise level and propagated interests are not only sent to all the hosts in its domain but also forwarded to the enterprise propagator.

Data received from the enterprise monitor is forwarded only to the host specified. Data received from a host in its domain is sent to the destination host if the specified host is in its domain else it is forwarded to the enterprise propagator. This may be slightly different if data coalescing is considered, in which case the data may have to be forwarded to multiple hosts.

12.3.3 Enterprise Propagator

An Enterprise propagator is at the top of the hierarchy overlooking all the domains. Its functionality is similar to but simpler than the domain propagator in the sense that it only has to keep track of the domains under it and forward data and interests between them based on their nature unless there are many enterprises under consideration.

13 Communication

The transmission of messages between entities is of considerable importance in distributed intrusion detection systems. The framework presented in this paper is based on the event-based communication model discussed by Hauswirth and Jazayeri [11], which is considered to be highly scalable. We term our communication model as an **interest-based** model as it incorporates the propagation of interests. As of now, we plan to adopt the communication mechanisms implemented in AAFID. But we intend to further investigate the suitability of the adopted communication mechanisms for intrusion detection systems in general and look for better alternatives.

14 Other Considerations

While we have discussed the main features of our framework, some other issues of considerable significance and interest are the following:

- **Security of agents**
  The ability of an agent to evoke responses from other agents by expressing an interest raises security concerns. For example, an intruder who is able to spoof interests might get access to some sensitive data about the hosts being monitored. There is also the possibility of denial-of-service attacks in which an attacker can generate a flood of interests to overwhelm the hosts. A mechanism to cancel interests might be needed to allow agents to stop receiving data that they are no longer interested in. But such a mechanism might also enable an attacker to spoof cancelation messages and prevent agents from receiving the data they need. These example scenarios suggest the need for privacy and authentication in our framework.

- **Clock Synchronization**
  Because a distributed intrusion detection system seeks to determine the time sequence of events resulting in an intrusion across multiple hosts, synchronization of clocks among the involved hosts is necessary. The degree of synchronization needed depends on the time granularity expected. The use of NTP [19] should be sufficient to meet such requirements.

- **Redundancy of Propagators**
  Propagators being lightweight modules may be replicated at different levels of the hierarchy to achieve load balancing and tolerance to failure and attack. Allowing communication between peer entities to avoid hierarchy whenever possible may also be an interesting possibility.

15 Conclusion

We have surveyed some of the existing approaches to communication mechanisms in intrusion detection systems, identified the deciding factors and the desired features of the communication mechanism for an IDS and looked at the distributed models of communication. And based on the motivation derived, we have presented a framework for distributed intrusion detection with agents being the only data analysis components. This framework uses an interest-based mechanism to transfer data between agents. This allows the agents to adapt to changing system state and also eliminates centralized analysis. Moreover, it has all the advantages of doing intrusion detection with agents as mentioned in AAFID [2] and the paper by Crosbie and Spafford [6].

We propose building a prototype based on this framework. We are interested in observing the impact of incorporating the interest-based mechanism on the size of the agents and on the host and network performance.

References

[1] S. Axelsson. Research in intrusion-detection systems: A survey. Technical report, Chalmers University of Technology, 1999.

[2] Jai Sundar Balasubramaniyan, Jose Omar Garcia-Fernandez, David Isacoff, Eugene Spafford, and Diego Zamboni. An architecture for intrusion detection using autonomous agents. In *Proceedings of the Fourteenth Annual Computer Security Applications Conference*, pages 13–24. IEEE Computer Society, December 1998.

[3] Joseph Barrus and Neil C. Rowe. A distributed autonomous-agent network-intrusion detection and response system. In *Proceedings of Command and Control Research and Technology Symposium, Monterey, CA*, pages 577–586, June 1998.

[4] Tim Bass. Multisensor data fusion for next generation distributed intrusion detection systems. In *Proceedings of the IRIS National Symposium on Sensor and Data Fusion*, May 1999. URL `http://www.silkroad.com/papers/html/iris/`.

[5] Matt Bishop. A standard audit trail format. In *Proceedings of the 1995 National Information Systems Security Conference, Baltimore, Maryland*, pages 136–145, October 1995.

[6] Mark Crosbie and Gene Spafford. Defending a computer system using autonomous agents. Technical Report 95-022, COAST Laboratory - Purdue University, 1994.

[7] T. Dierks and C. Allen. The TLS protocol version 1.0. RFC 2246, January 1999.

[8] D. Estrin, R. Govindan, J. Heidemann, and S.Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of Mobicom'99, Seattle, Washington*, pages 263–270, 1999.

[9] R. Feiertag, C. Kahn, P. Porras, D. Schnackenberg, S. Staniford-Chen, and B. Tung. A common intrusion specification language (CISL). http://www.gidos.org/drafts/language.txt. June 2000.

[10] D. Gupta, T.C. Buchheim, B.S. Feinstein, G.A. Matthews, and R.A. Pollock. IAP: Intrusion Alert Protocol. Internet-Draft, February 2001. URL `http://www.ietf.org/internet-drafts/draft-ietf-idwg-iap-04.txt`.

[11] M. Hauswirth and M. Jazayeri. A component and communication model for push systems. In *Proceedings of ESEC/FSE 99 - Joint 7th European Software Engineering Conference (ESEC) and 7th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE-7), Toulouse, France*, September 1999.

[12] S. Hofmeyr. *An Immunological Model of Distributed Detection and Its Application to Computer Security*. PhD thesis, University of New Mexico, May 1999.

[13] Dennis J. Ingram. Autonomous agents for distributed intrusion detection in a multi-host environment. Master's thesis, Naval Postgraduate School, Monterey, CA, September 1999.

[14] Y. Frank Jou, Fengmin Gong, Chandru Sargor, Shyhtsun Felix Wu, and W. Rance Cleaveland. Architecture design of a scalable intrusion detection system for the emerging network infrastructure. Technical Report CDRL A005, Research Triangle Park, N.C. 27709, 1997.

[15] C. Kahn, Don Bolinger, and Dan Schnackenberg. Communication in the common intrusion detection framework v 0.7. Draft Specification, June 1998. URL `http://www.gidos.org/drafts/communications.txt`.

[16] C. Kahn, P. Porras, S. Staniford-Chen, and B. Tung. A common intrusion detection framework. In *Journal of Computer Security*, 1998.

[17] G. Mansfield and D. Curry. Intrusion detection message exchange format: Comparison of SMI and XML implementations. Internet Draft, 2000. URL `http://www.ietf.org/internet-drafts/draft-ietf-idwg-xmlsmi-01.txt`.

[18] Peter Mell and Mark McLarnon. Mobile agent attack resistant distributed hierarchical intrusion detection system. In *Proceedings of RAID'99, CERIAS, Purdue University*, 1999.

[19] D. Mills. Network time protocol (version 3). RFC 1305, March 1992.

[20] Peng Ning, X. Sean Wang, and Sushil Jajodia. Modeling requests among cooperating intrusion detection systems. *Computer Communications*, 23(17):1702–1715, 2000.

[21] Peng Ning, X. Sean Wang, and Sushil Jajodia. A query facility for common intrusion detection framework. In *Proceedings of the 23rd National Information Systems Security Conference, Baltimore, MD*, pages 317 – 328, Oct 2000.

[22] C. Partridge and R. Hinden. Version 2 of the Reliable Data Protocol (RDP). RFC 1151, April 1990.

[23] Vern Paxson. Bro: A system for detecting network intruders in real-time. In *Proceedings of the 7th Annual USENIX Security Symposium*, San Antonio, TX, January 1998. URL `ftp://ftp.ee.lbl.gov/papers/bro-usenix98-revised.ps.Z`.

[24] Phil Porras, Dan Schnackenberg, Stuart Staniford-Chen, Davis, Maureen Stillman, and Felix Wu. The common intrusion detection framework architecture. Document. URL `http://www.gidos.org/drafts/architecture.txt`.

[25] Phillip A. Porras and Peter G. Neumann. EMERALD: event monitoring enabling responses to anomalous live disturbances. In *1997 National Information Systems Security Conference*, Oct 1997.

[26] Thomas H. Ptacek and Timothy N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, Secure Networks, Inc., January 1998.

[27] S. Snapp, J. Brentano, and G. Dias et al. DIDS (Distributed Intrusion Detection System) – motivation, architecture, and an early prototype. In *Proceedings of the 14th National Computer Security Conference*, October 1991.

[28] Eugene H. Spafford and Diego Zamboni. Intrusion detection using autonomous agents. *Computer Networks*, 34 (4):547–570, October 2000.

[29] W. Stallings. *SNMP, SNMP v2, SNMP v3, and RMON 1 and 2*. Addison-Wesley, third edition, 1999.

[30] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS-a graph based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, September 1996.

[31] Jiahai Yang, Peng Ning, X. Sean Wang, and Sushil Jajodia. CARDS: A distributed system for detecting coordinated attacks. In *Proceedings of IFIP TC11 Sixteenth Annual Working Conference on Information Security*, pages 171 – 180, Aug 2000.

[32] Diego Zamboni. Doing intrusion detection using embedded sensors. Technical Report 2000-21, CERIAS, Purdue Univesrity. URL `https://www.cerias.purdue.edu/techreports-ssl/public/2000-21.pdf`. Thesis Proposal.