

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

**Rajeev Gopalakrishna, Eugene H. Spafford**

Center for Education and Research in Information Assurance and Security,  
1315 Recitation Building, Purdue University, West Lafayette, IN 47907-1315, USA  
{rgk,spaf}@cerias.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. In this paper, 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

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 [14], GrIDS [16], EMERALD [13] and AAFID [1, 15]. Spafford and Zamboni [15] 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.

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 [7]. 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.

## 2 Previous Work

DIDS [14] is a distributed intrusion detection system consisting of host managers and LAN managers doing distributed data monitoring and sending notable events to the DIDS director. They also do some local detection, passing the summaries 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.

GrIDS [16] 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.

EMERALD [13] 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.

AAFID [1, 15] 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. [11, 12] recognize the importance of the querying facility in cooperative intrusion detection systems. They propose an extension to the common intrusion specification language (CISL) [5] to allow intrusion detection components to specify requests for particular information from other components.

In the paper by Crosbie and Spafford [3], 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 [2] and Ingram [8], 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 [9] 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 [17] 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" [12]. 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.

### **3 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 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.

### **4 Agents**

The definition and advantages of autonomous agents are discussed in detail in AAFID [1] and the paper by Crosbie and Spafford [3]. 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 [1, 15].

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

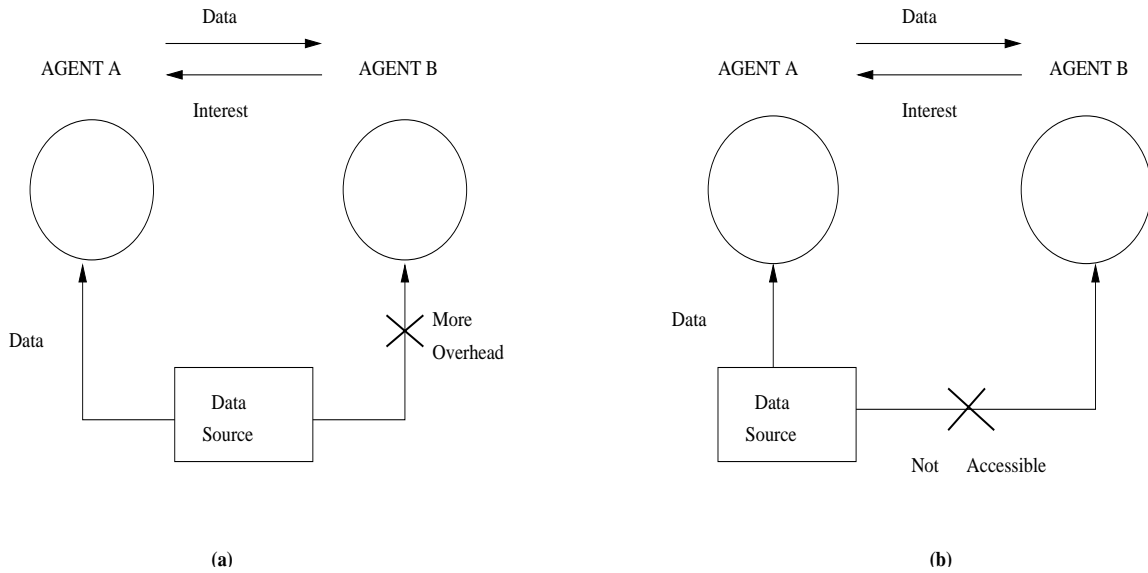


Figure 1: Generation of an interest based on locality of data collection. (a) To minimize overhead. Agent B can collect the data itself, but the overhead of obtaining it is larger than requesting it from Agent A. (b) Inaccessibility of data. Agent B does not have access to the data it needs, so it requests it from Agent A which has access to the data.

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.

## 5 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. 1(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. 1(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.

## 5.1 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. [4] which looks at scalable coordination of sensors running localized algorithms and using diffusion for interest propagation to achieve a global result. 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.

## 5.2 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:

### 5.2.1 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.

### **5.2.2 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.

### **5.2.3 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 be 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. 1(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. 1(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.

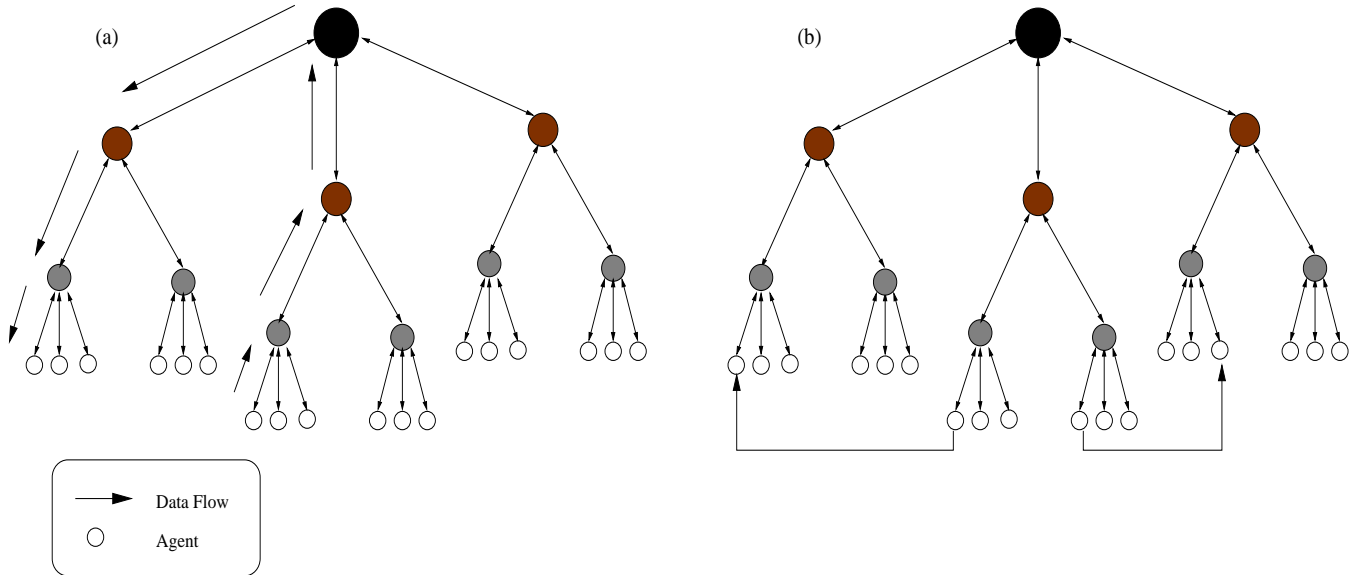


Figure 2: Delivery of data to the agent expressing an interest (a) Using the path through the hierarchy (b) Using a direct connection

### 5.3 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.

## 6 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. 2(a)).
- Data can be delivered to the agent directly without going through the hierarchy (see Fig. 2(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.

- **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.

## 7 Component Description

In this section, we describe the components of the architecture as shown in figures 3 and 4:

### 7.1 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.

### 7.2 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.

### 7.3 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

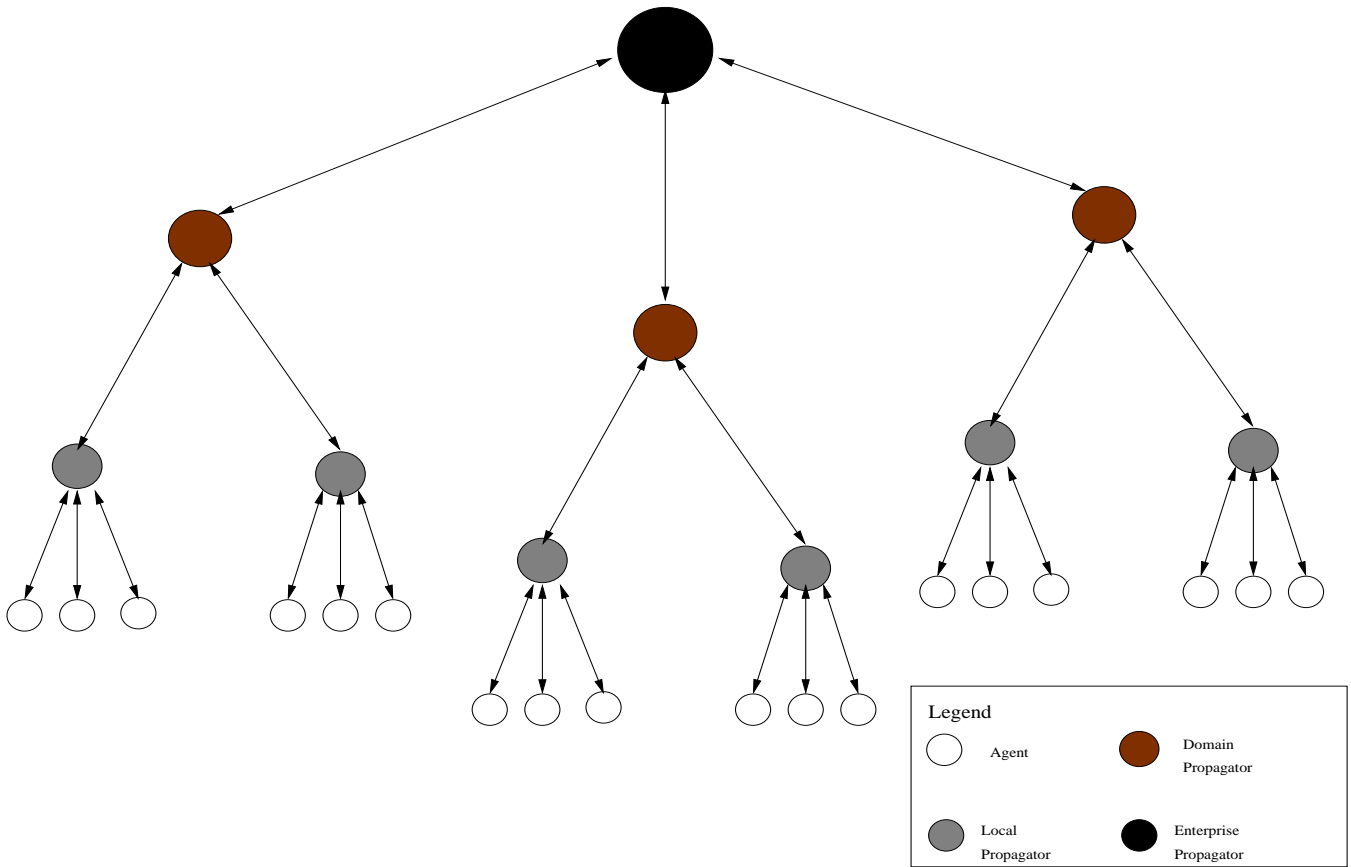


Figure 3: Logical organization of agents and propagators showing the communication hierarchy. The bidirectional arrows represent the flow of both interests and data between the entities.

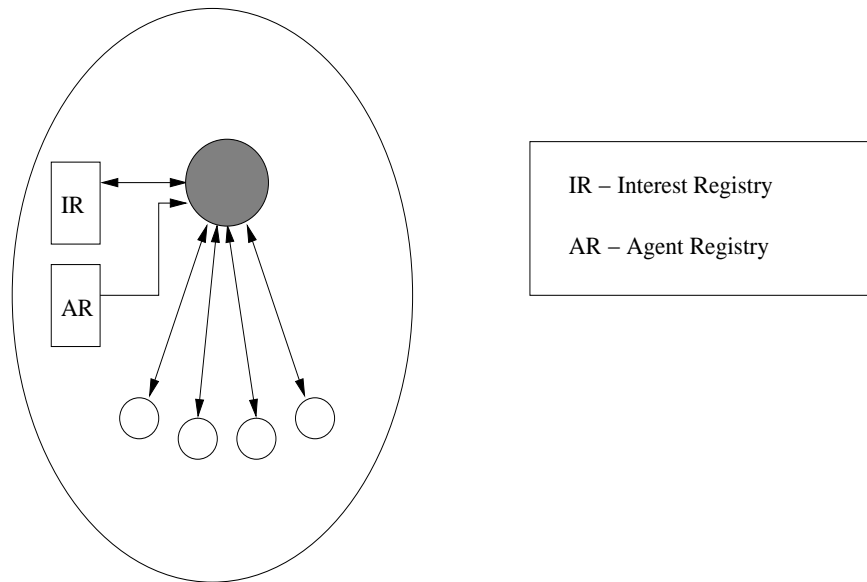


Figure 4: Physical layout of the components in a host showing agents, interest and agent registries and the local propagator

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 2. 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 2 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:

### **7.3.1 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.

### **7.3.2 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.

### 7.3.3 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.

## 8 Communication Mechanism

The transmission of messages between entities is of considerable importance in distributed intrusion detection systems. The communication mechanism needs to be reliable, efficient and secure and the communication model as a whole should be scalable. The paper on AAFID [1] has a good summary of the options available for intra-host and inter-host communication and Hauswirth and Jazayeri [6] compare the distributed communication models. The framework presented in this paper is based on the event-based communication model discussed by Hauswirth and Jazayeri [6], which is considered to be highly scalable. 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. An example is the use of TCP connections between the components of the intrusion detection system. While TCP provides reliability, it also has certain features such as congestion control and slow start, which do not seem to be ideal for a system such as an intrusion detection system. The alternative appears to be the RUDP (Reliable UDP) protocol, which has all the features of TCP except congestion control and slow start.

## 9 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 cancellation 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 [10] 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.

## 10 Conclusion

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 [1] and the paper by Crosbie and Spafford [3].

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] Jai Sundar Balasubramanian, 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.
- [2] 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.
- [3] Mark Crosbie and Gene Spafford. Defending a computer system using autonomous agents. Technical Report 95-022, COAST Laboratory - Purdue University, 1994.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] S. Hofmeyr. *An Immunological Model of Distributed Detection and Its Application to Computer Security*. PhD thesis, University of New Mexico, May 1999.

- [8] Dennis J. Ingram. Autonomous agents for distributed intrusion detection in a multi-host environment. Master's thesis, Naval Postgraduate School, Monterey, CA, September 1999.
- [9] Peter Mell and Mark McLarnon. Mobile agent attack resistant distributed hierarchical intrusion detection system. In *Proceedings of RAID'99, CERIAS, Purdue University*, 1999.
- [10] D. Mills. Network time protocol (version 3). RFC 1305, March 1992.
- [11] Peng Ning, X. Sean Wang, and Sushil Jajodia. Modeling requests among cooperating intrusion detection systems. *Computer Communications*, 23(17):1702–1715, 2000.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] Eugene H. Spafford and Diego Zamboni. Intrusion detection using autonomous agents. *Computer Networks*, 34 (4):547–570, October 2000.
- [16] 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.
- [17] 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.