Applying Mobile Agents to Intrusion Detection and Response

Wayne Jansen, Peter Mell, Tom Karygiannis, Don Marks
National Institute of Standards and Technology
Computer Security Division

NIST Interim Report (IR) - 6416
October 1999

Table of Contents

1. Introduction

This report is an initial foray into the relatively unexplored terrain of using Mobile Agents (MAs) for Intrusion Detection Systems (IDSs). It is a research guide that helps identify the most promising areas of mobile agent IDS research. After providing some background information, we enumerate the problems found in current IDSs and propose potential solutions offered by MAs. The report suggests innovative ways to apply agent mobility to address shortcomings of current IDS designs and implementations. We then discuss performance advantages and disadvantages that occur when using MAs for IDSs. The practical discussion of performance leads into proposals for several new intrusion detection paradigms enabled by MAs. While the report focuses mostly on the benefits derived from mobility, we also take into consideration the features gained from agent technology, such as autonomous components, which offer significant benefits. We explore these benefits in some detail and propose specific research topics in both the intrusion detection and intrusion response areas.

1.1. Background

Intrusion Detection Systems (IDSs) are proliferating throughout corporate, government, and academic computer networks. It is estimated that sales of IDS tools reached $100 million in 1998 [BACE99]. Intrusion detection is not an emerging research field. It is a well-established commercial area with several large competitors like Cisco and Network Associates[1]. Admittedly, the IDS products themselves produce many false positives and do not detect all known attacks. However, the development of IDS products is likely to parallel the past development of anti-virus software. Original anti-virus software created an alarm every time users created new files. However, over several years the anti-virus software has progressed to the point that few users even notice that the anti-virus software is running and they have confidence that it detects all known viruses.

The concept of creating an intrusion detection system was first proposed in 1980 by James Anderson [ANDE80]. However, the field did not take off until 1987 when Dorothy Denning published an intrusion detection model [DENN87]. In 1988, at least three IDS prototypes were created [BAUE88] [SEBR88] [SMAH88]. In the following years, an ever-increasing number of research prototypes were explored. The US government, realizing that its computer systems were insecure, provided significant funding for research in IDSs. Hundreds of millions of dollars have probably been spent on IDS research within the last ten years.

Because intrusion detection has become a mature industry and a proven technology, nearly all of the easy problems have been solved. No major breakthroughs in intrusion detection research have recently been made. Instead, commercial companies are mostly perfecting existing intrusion detection techniques. With the maturation of the intrusion

[1] Certain computer manufacturers' products and standards are discussed in this paper. The discussion is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the products and standards identified are necessarily the best available.

detection field, traditional lines of intrusion detection research are having diminishing returns. Therefore, future intrusion detection research is expected to focus on relatively unexplored areas such as:

- Attack response mechanisms,
- Architectures for highly distributed intrusion detection systems,
- Intrusion detection inter-operability standards, and
- New paradigms for performing intrusion detection.

1.2. Mobile Agent Technology

IDSs implemented using MAs is one of the new paradigms for intrusion detection. MAs are a particular type of software agent, having the capability to move from one host to another. A software agent can be defined as [BRAD97]:

> "… a software entity which functions continuously and autonomously in a particular environment … able to carry out activities in a flexible and intelligent manner that is responsive to changes in the environment … Ideally, an agent that functions continuously … would be able to learn from its experience. In addition, we expect an agent that inhabits an environment with other agents and processes to be able to communicate and cooperate with them, and perhaps move from place to place in doing so."

Mobile agents have been a research topic of interest for several years, yet this research has for the most part remained within laboratories and has not experienced a wide-scale adoption by industry. The development of the World Wide Web application, however, has dramatically stimulated interest in this area of research by offering the possibility of a widely deployed application that could use mobile agent technology. The research community visualizes mobile agents launched via web browsers to gather information and interact with any node in the network. IBM and General Magic were early pioneers of this vision, [CHES95, HARR95]. Concurrent with this effort, ARPA sponsored a Knowledge Sharing program. The KQML language [FINI94] was developed under this program and remains one of the viable Agent Communication Languages (ACLs).

This research area was reformulated in the '95-'96 time frame when Java was released by Sun Microsystems. Although Java is simply a new interpreted computer language, it is designed for network interactions and is a powerful enabling technology for mobile code. Web browsers were quickly "Java-enabled" and the IT community seemed convinced that mobile code would quickly become a reality.

The Java language provided some system independence and considerable security features were included in the language and implementations. These are not unique features, of course, they simply were implemented better in Java than other languages and so Java became extremely popular. During this same period, numerous proposals for mobile agent implementations were fielded. For example, the Lava system [WU96, HANS97] was developed at North Carolina State University. This system focused on

security problems and developed a simple security policy for applets. Mitre Corporation [FARM96, FARM97] also pursued work in this area, developing authentication mechanisms and defining a taxonomy of security related problems.

An important observation to make about most of the early work in this field is the assumption made by most researchers about a totally open system. That is, the security problems being addressed are those found in a system with open connectivity and with the maximum possible threats. Several researchers reached conclusions indicating that the paradigm was not useful since there were always certain threats that could not be adequately countered while maintaining a totally open system.

Partly because of these conclusions, as well as well publicized attacks against early Java-enabled systems, security related problems have hindered the widespread adoption of MA technology. Security architectures have been defined, but they contain too much residual risk for most applications. Recent work at the University of Tulsa, for example, proposes using mobile agents for data mining purposes. Such an application requires providers of information to keep their systems "open" to a multitude of users, most of whom are unknown to the host. A good overview of current mobile agent projects and technology is provided in [MARR98].

However, relatively little work has been done on using a mobile agent architecture for the purpose of providing a security capability, such as intrusion detection. If a mobile agent architecture is designed for a specific purpose such as system administration or security function maintenance, then strong authentication may be enforced and the residual risk decreases significantly.

While MAs are an extraordinarily powerful tool, their implementation has been hindered by security considerations. These security considerations are especially critical for intrusion detection systems, with the result that most security research in this field has concentrated upon the architecture necessary to provide security for mobile agents. We claim that such negative results are not fatal to the proposed study since these security issues are likely to be addressed by the research community and there will be few authorized users of the MA-based IDSs within an organization.

1.3. Related Work

Current work in applying agents to intrusion detection is being conducted at a number of research labs, including the University of Idaho, the University of New Mexico, the Army Research Laboratory, Iowa State University, Purdue University, and the Information-technology Promotion Agency in Japan. The existence of other work in this field is encouraging, however, existing research lacks the scope necessary for successful integration of the two fields. In many cases, these projects are in preliminary stages, don’t directly involve intrusion detection, or don’t use the mobility aspect of agents. Thus, the current work is exciting, but not complete, and supports a more thorough investigation of Mobile Agent Intrusion Detection Systems (MAIDS).

1.3.1. Autonomous Agents for Intrusion Detection

The Autonomous Agents for Intrusion Detection (AAFID) effort at Purdue [BALA98] is in many ways a classical IDS with agents used mainly as a means for structuring the intrusion detection collection component into a set of lightweight software components which can be easily reconfigured. AAFID employs a hierarchy of agents. At the root of the hierarchy are monitors, which provide global command and control and perform analysis of information flowing from lower level nodes. At the leaves are agents that collect event information. The agents reside on special purpose agent platforms, called transceivers. Transceivers perform command and control of locally running agents and the analysis or reduction processing of the information received from the agents. Transceivers feed processed information onto monitors. Agents appear to be static once they are deployed to a transceiver, but are also replaceable through reconfiguration.

1.3.2. Hummingbird

The University of Idaho has developed the Hummingbird project [FRIN98]. This is one of the more ambitious distributed intrusion detection prototypes available. The Hummingbird system is a distributed system for managing misuse data. While the system uses some agent technology, the agents are not autonomous, nor are they mobile. Only the data collection is distributed and control remains centralized. Emphasis is placed on sharing security relevant data among sites having different security domains. The tools, algorithms, data reduction and visualization techniques offer considerable promise for use in a mobile agent system. Hummingbird does not implement new security features to protect itself. Instead, it relies on the Kerberos [NEUM94] system.

1.3.3. Java Agents for Meta-Learning

The Java Agents for Meta-Learning (JAM) project [LEE99] at Columbia University, NY, applies meta-learning to distributed data mining, using intelligent agents. Intelligent agents employ artificial intelligence techniques to model knowledge and reasoning, as well as behavior, in multi-agent societies or domains. The design has two key components: local fraud detection agents that learn how to detect fraud and provide intrusion detection services within a single corporate information system, and a secure, integrated meta-learning system that combines the collective knowledge acquired by individual local agents. Data mining, like neural networks and other single-point learning applications, does not enable knowledge sharing among agents. The meta-learning approach attempts to overcome this limitation by integrating a number of separately learned classifiers embodied as remote agents.

1.3.4. Intelligent Agents for Intrusion Detection

This project at Iowa State University [HELM98] involves an IDS based upon intelligent agent technology, in a manner somewhat similar to JAM. Agent mobility allows various types of intelligent agents that employ classifier algorithms to travel among collection points, referred to as “data cleaners,” and uncover suspicious activities. The agent

algorithms are standard sequence identification methods and feature vector identification detection mechanisms. The architecture is hierarchical, with a data warehouse at the root, data cleaners at the leaves, and classifier agents in between. A classifier agent specializes on a specific category of intrusion and is capable of collaborating with agents of another category to determine the severity level of an activity deemed suspicious. Moving the computational analysis (i.e., the classifier agent) to each collection point, avoids the costly movement of information to an aggregation unit. The results provide a good basis for follow-on work, since the approach establishes the capability to define intrusion detection agents that target individual systems and subsystems.

1.3.5. Advanced Telecommunications/Information Distribution Program

Work being done by the Army on the Advanced Telecommunications/Information distribution Research Program (ATIRP) [CONN99, JACO99, BARR98] addresses the detection computer vulnerabilities using MAs, not intrusion detection. However, intrusion detection modules could easily be substituted for vulnerability assessment modules to create a rudimentary IDS. A central dispatcher launches agents to one or more target nodes to test for known vulnerabilities and report back results. Agents are composed dynamically using a genetic algorithm, which continually attempts to maximize the likelihood of discovering existing vulnerabilities. The gene pool from which agents evolve consists of code fragments that correspond to a detection technique and have been designed for composition with other fragments. The architecture has significant security capabilities and is based upon cryptographic signatures and public key certificates. Identical or similar capabilities would be required in an IDS. The system would have to be expanded to manage an IDS, since inter-agent communication is more critical in intrusion detection than in vulnerability scanning.

1.3.6. Intrusion Detection Agent System

The Information-technology Promotion Agency (IPA) in Japan, is developing an IDS called the Intrusion Detection Agent system (IDA) [ASAK99]. The IDA is a multi-host-based IDS. Instead of analyzing all of the users' activities, IDA works by watching specific events that may relate to intrusions, referred to as Marks Left by Suspected Intruder (MLSI). If an MLSI is found, IDA gathers information related to the MLSI, analyzes the information, and decides whether or not an intrusion has occurred.

The IDA system relies on mobile agents to trace intruders among the various hosts involved in an intrusion and to gather information. The architecture is hierarchical, with a central manager at the root and a variety of agents at the leaves. A sensor is an agent that resides at a node in search of MLSIs. Upon discovery of such information, the sensor notifies the manager who dispatches a tracing agent to the host. The tracing agent initiates an information-gathering agent to collect related information at the host, before moving onto any other site identified as a suspected point of origin. The manager collects and integrates the results from the information-gathering agent as they return. Possible duplication caused by multiple sensors detecting the same intrusion is resolved through a

message board at each monitored host. The developers indicate that the resulting multi-agent system is an efficient and effective way for detecting intrusions.

2. IDS Requirements

At least one past effort has identified desirable characteristics for an IDS. In [CROS95], the authors indicate that, regardless on what mechanisms an IDS is based, it must do the following:

- Run continuously without human supervision,
- Be fault tolerant and survivable,
- Resist subversion,
- Impose minimal overhead,
- Observe deviations from normal behavior,
- Be easily tailored to a specific network,
- Adapt to changes over time, and
- Be difficult to fool.

We have developed a similar set of requirements along two themes: functional and performance requirements.

2.1. Functional Requirements

As the network-computing environment increases in complexity, so do the functional requirements of IDSs. Common functional requirements of an IDS being deployed in current or near-term operational computing environments (see Appendix A for more information on the operational environments envisioned) include the following:

- The IDS must continuously monitor and report intrusions.

- The IDS must supply enough information to repair the system, determine the extent of damage, and establish responsibility for the intrusion.

- The IDS should be modular and configurable as each host and network segment will require their own tests and these tests will need to be continuously upgraded and eventually replaced with new tests.

- Since the IDS is assigned the critical role of monitoring the security state of the network, the IDS itself is a primary target of attack. The IDS must be able to operate in a hostile computing environment and exhibit a high degree of fault-tolerance and allow for graceful degradation.

- The IDS should be adaptive to network topology and configuration changes as computing elements are dynamically added and removed from the network.

- Anomaly detection systems should have a very low false alarm rate. Given the projected increase in network connectivity and traffic, simply decreasing the

percentage of overall false alarms may not be sufficient as their absolute number may continue to rise.

- The IDS should be able to learn from past experiences and improve its detection capabilities over time. A self-tuning IDS will be able to learning from false alarms with the guidance of system administrators and eventually on its own.

- The IDS should be able to be easily and frequently updated with attack signatures as new security advisories and security patches become available and new vulnerabilities and attacks are discovered.

- Decision support tools will be necessary to help system administrators respond to various attacks. The IDS will be required not only to detect anomalous events, but also to take automated corrective action.

- The IDS should be able to perform data fusion and be able to process information from multiple and distributed data sources such as firewalls, routers, and switches. As real-time detection demands push networked-based solutions to re-programmable hardware devices that can download new capabilities, the IDS will need to be able to communicate with the hardware-based devices.

- Data reduction tools will be necessary to help the IDS process the information gathered from data fusion techniques. Data mining tools will be helpful in running statistical analysis tools on archived data in support of anomaly detection techniques.

- The IDS should be capable of providing an automated response to suspicious activity. Rapid changes in network conditions and limited network administration expertise make it difficult for system administrators to diagnose problems and take corrective action to minimize the damage that intruders can cause.

- The ability to detect and react to distributed and coordinated attacks will become necessary. Coordinated attacks against a network will be able to marshal greater forces and launch many more and varied attacks against a single target. These attacks can be permutations of known attacks, be rapidly evolving, and be launched at little cost to the attackers.

- Distributing the computational load and the diagnostic capabilities to agents scattered throughout the network adds a level of fault-tolerance, but it is often necessary for the system administrator to have control over the IDS from a central location.

- The IDS should be able to work with other Commercial Off-the-Shelf (COTS) security tools, as no vendor toolset is likely to excel in or to provide complete coverage of the detection, diagnosis, and response responsibilities. The IDS framework should be able to integrate various data reduction, forensic, host-based, and network-based security tools. Interoperability and conformance to standards will further increase the value of the IDS.

- IDS data often requires additional analysis to assess any damage to the network after an intrusion has been detected. Although the anomalous event was the first detected, it may not be the first attempt to gain unauthorized access to the network. Post event analysis will be needed to identify compromised machines before the network can be restored to a safe condition.

- The IDS itself must also be designed with security in mind. For example, the IDS must be able to authenticate the administrator, audit administrator actions, mutually authenticate IDS devices, protect the IDS data, and not create additional vulnerabilities.

2.2. Performance Requirements

An IDS that is functionally correct, but that detects attacks too slowly is of little use. Thus we must enumerate several performance requirements for IDSs. The IDS performance requirements include:

- To the extent possible, anomalous events or breaches in security should be detected in real-time and reported immediately to minimize the damage to the network and the loss or corruption of confidential information.

- The IDS must not place undue burden or interfere with the normal operations for which the systems were bought and deployed to begin with. This requirement makes it necessary for the agents to be cognizant of the consumption of network resources for which they are competing. There is a tradeoff between additional levels of security monitoring and the performance penalty to be paid by other applications.

- The IDS must be scalable. As new computing devices are added to the network, the IDS must be able to handle the additional computational and communication load.

3. Mobile Agents for Intrusion Detection

For mobile agents to be useful for intrusion detection, it is necessary that many, if not all, hosts and network devices are installed with an MA platform. This is not a far-fetched assumption because an MA platform is general-purpose software that enables organizations to implement many different applications. If MAs become popular, every new host may come preinstalled with a MA platform just as today most personal computers come bundled with a Java interpreter in the web browser. Contrast this to many IDS schemes that assume that a host-based IDS is installed on every host. It is generally too expensive to install a proprietary solution (like a host-based IDS) on every host in a network, but it is not unusual to install a general-purpose interpreter (like an MA platform and Java virtual machine) on every host.

3.1. Advantages

A number of advantages of using mobile code and mobile agent computing paradigms have been proposed [LANG98, SMIT88]. These advantages include: overcoming network latency, reducing network load, executing asynchronously and autonomously, adapting dynamically, operating in heterogeneous environments, and having robust and fault-tolerant behavior. This section examines these claims and evaluates their applicability to the design of ID systems.

3.1.1. Overcoming Network Latency

Mobile agents are useful for applications that need to respond in real time to changes in their environment, because they can be dispatched from a central controller to carry out operations directly at the remote point of interest. In addition to detecting and diagnosing potential network intrusions, an IDS needs to provide an appropriate response in order to protect and defend the network from malicious behavior. While a central controller can send messages to the nodes within the network and issue instructions on how to respond to a particular condition or perceived threat, the approach is problematic. For example, the central controller may have to respond to a number of events throughout the network in addition to handling its normal processing load and become a bottleneck or a single point of failure. If connections to this central server are slow or unreliable, the network communications are susceptible to unacceptable delays. Mobile agents, since they are distributed throughout the network, may take advantage of alternate routes around any problem communication links.

It will always be faster to send a message to a network node to execute predetermined, resident code, rather than send a mobile agent to the node. However, such an architecture requires that all response and reconfiguration actions be predefined, replicated and distributed throughout the network. The response mechanism then constitutes, in effect, a large distributed database, raising serious administration problems concerning configuration management, consistency and transaction control. Innovative responses, by definition, must be transmitted at least once to each affected node, either by conventional

network means, a series of messages, or by a mobile agent. Of these choices, the mobile agent technique offers the fastest response.

3.1.2. Reducing Network Load

One of the most pressing problems facing current IDSs is the processing of the enormous amounts of data generated by the network traffic monitoring tools and host-based audit logs. IDSs typically process most of this data locally. However, abstracted forms of the data are often sent to other network locations where the data is further abstracted and then eventually sent to a central processing site that evaluates abstracted results from all location in the network. Even though the data is usually abstracted before being sent out on the network, the amount of data can still place a considerable communication load on the network. Mobile agents offer an opportunity to reduce the network load by eliminating the need for this data transfer.

Mobile agents are well suited for *ad hoc*, flexible, search and analysis problems involving multiple distributed resources that require specialized tasks that are not supported by the data server. A mobile agent-based search and data analysis approach can help decrease network traffic resulting from the transfer of large amounts of data across a network for local processing. Instead of transferring the data across the network, mobile agents can be dispatched to the machine on which the data resides, essentially moving the computation to the data, instead of moving the data to the computation, thus reducing the network load for such a scenario. Clearly, transferring an agent that is smaller in size than the data to be transferred reduces the network load. These benefits hold when the comparison is made between encrypted lightweight mobile agents and the relatively larger data to be transferred.

3.1.3. Asynchronous Execution and Autonomy

IDS architectures that are coordinated by a central host require reliable communication paths to the network sensors and intermediate processing nodes. The critical role played by this central controller makes it a likely target of attack. Mobile agent frameworks allow IDSs to continue operation in the event of the failure of a central controller or communication link. Unlike message passing routines or Remote Procedure Call (RPC), once the mobile agent is launched from a home platform it can continue to operate autonomously even if the host platform from where it was launched is no longer available or connected to the network. The coordination of IDS sensors and filters can be protected from the loss of network connections since the mobile agents do not require control by another process. A mobile agent's inability to communicate with central controller would not prevent it from carrying out its assigned tasks.

Although disconnected operation is possible, a number of issues need to be addressed. Distributing the functions of a central controller among the network components is a non-trivial problem. Another problem concerns the operational methods of MAs themselves. For example, Java-based MAs typically load their class files dynamically, as needed, from their home platform. The ability to dynamically load classes also has security

implications. If the home platform is not available, these class files may be provided by the local host or must be found and transferred from a remote trusted host. Class loading from a remote platform or the local host platform raises a number of security issues. The class files may have been modified in such a way as to alter the functionality of the agent or even to allow for eavesdropping of the agents' transactions. Class versioning problems may also yield problems from which the MAs may be able to recover.

3.1.4. Structure and Composition

MAs allow for a natural way to structure and design an IDS. For example, rather than a monolithic static system, an IDS can be divided into data producer and data analyzer components and represented as agents. The data producer provides an interface to the networks it sniffs or audit trails it filters. Multiple analyzers, each responsible for detecting a single attack or a small set of attacks, interact with the producer to look for attacks. Under such a framework, MAs from multiple vendors can be used to create an IDS. If a company has the best detector for attack X and another company has the best detector for attack Y, then we can use MAs from both vendors to detect X and Y. Even where manufacturers do not produce agent-based products, it may be possible to reconstitute the product as an agent through wrapping or other techniques. In such an environment, users can also write customized MAs to detect events specific to their environment and work seamlessly with the other MA components. Although this approach applies equally as well to an IDS composed of static components, the agent orientation and mobility considerations provide inherent motivation for identifying and compartmentalizing functionality.

3.1.5. Adapting Dynamically

Just as the network's configuration, topology, and traffic characteristics change over time, so should the types of network tests performed and activities monitored. Each computing node in the network will require different tests and these tests will change over time; some tests will no longer be necessary, while new tests will need to be added to the test suite as new vulnerabilities and threats evolve. MAs provide a versatile and adaptive computing paradigm as they can be retracted, dispatched, cloned, or put to sleep as network and host conditions change. For example, as better MAs detectors for an attack are developed they can be sent out on the network to replace the older version, or if an MA is producing too many false positives it can be recalled or gracefully terminated.

MAs also have the ability to sense their execution environment and autonomously react to changes. For example, if the computational load of the host platform is too high and the host's performance doesn't meet the agent's service expectations, the agent and its data can move to another machine that can better satisfy its computational needs. MAs can distribute themselves among the hosts in the network in such a way as to maintain the optimal configuration for solving a particular problem.

3.1.6. Operating in Heterogeneous Environments

Large enterprise networks are typically comprised of many different computing platforms and computing devices. One of the greatest benefits of MAs is the implementation of interoperability at the application layer. Interoperability at the computer or transport layer, such as solutions provided by single vendors, requires significant changes to the host's environment. Interoperability at the presentation layer, such as the CIDF model [CIDF], limits flexibility in updating the system for new attacks. Conversely, while MA frameworks must be installed on each host, MAs themselves are independently configurable. Since mobile agents are generally computer and transport-layer independent, and dependent only on their execution environment, they offer an attractive approach for heterogeneous system integration. MAs' ability to operate in heterogeneous computing environments is made possible by a virtual machine or interpreter on the host platform. Data fusion efforts can be facilitated by having mobile agents run on switches, routers, and other networking elements. MAs can run on any computing node that can host an agent platform. The ability of MAs to operate in heterogeneous environments also provides an opportunity for the easy integration of network-based and host-based tools operating on various platforms. COTS interoperability may also be facilitated via the use of Agent Communication Languages (ACL) designed for network security testing and intrusion detection domains. Although the MA framework allows an IDS to operate in heterogeneous environments, the tests performed or tasks assigned to the mobile agents are for the most part platform-dependent. Therefore, unless a common programming interface for intrusion detection functions is available, agents must either be restricted to a single class of host or be designed to accommodate heterogeneity in some fashion (e.g., dynamically load or intrinsically convey the host dependent code).

3.1.7. Robust and Fault-tolerant Behavior

The ability of mobile agents to react dynamically to unfavorable situations and events makes it easier to build robust distributed systems. For example, if a host is being shut down, all agents executing on that machine are warned, whenever possible, and given time to dispatch and continue their operation while preserving their execution state on another host in the network. Their support for disconnected operation and distributed design paradigms eliminate single point of failure problems and allow mobile agents to offer fault-tolerant characteristics. While there are many features of MAs that enable applications to be robust and fault-tolerant, we should mention a few drawbacks.

The ability of the mobile agents to move from one platform to another in a heterogeneous environment has been made possible by the use of virtual machines and interpreters. Virtual machines and interpreters, however, can offer only limited support for preservation and resumption of the execution state in heterogeneous environments because of differing representations in the underlying hardware. For example, the full execution state of an object cannot be retrieved in Java. Information such as the status of the program counter and frame stack is currently forbidden territory for Java programs.

Conventional fault-recovery techniques aren't sufficient for the mobile agent computing paradigm. For example, checkpointing before and after arrival, and upon completion of certain transactions or events may be necessary to ensure for acceptable fault-recovery. With each check-pointing procedure and non-repudiation mechanism invoked, however, more overhead is introduced. Even though an arsenal of techniques exist to provide security and fault-tolerance, the designer must be careful in selecting which mechanisms to use and how they impact the overall system performance and functionality.

Although mobile agents possess a great deal of autonomy and perform well in disconnected operations, the failure of the home platform or other platforms that the agents rely on to provide security services can seriously reduce their intended functionality. Even though a mobile agent can become more fault-tolerant by moving to another machine, the mobile agent's reliance on the safe operation of a safe home or trusted platform places restrictions on its functionality. Designers of mobile agent platforms are also faced with tradeoffs between security and fault-tolerance. For example, in order to address the security risks involved in "multi-hop" agent mobility, some agent architectures have been built on centralized client-server models requiring agents to return to a central server before moving on to another host machine [JUMPIN]. Clearly, addressing the security risks in this manner renders all the mobile agents vulnerable to a failure of the central server.

3.1.8. Scalability

The computational load on centralized IDSs increases as more processing nodes are added to the networks they monitor. As networking technology continues to improve, increased bandwidth and network traffic will place greater demands on these centralized architectures. Distributed MA IDS architectures are one of several options that allow computational load and diagnostic responsibilities to be distributed throughout a network. As the number of computing elements in the network increases, agents can be cloned and dispatched to new machines in the network.

3.2. Disadvantages

The obvious disadvantage of using MAs is the concern that they will introduce vulnerabilities into the network. However, this is not the only disadvantage to implementing Mobile Agent Intrusion Detection System (MAIDS). MA solutions may not perform fast enough to meet the IDS’s needs. In addition, the MAs may contain large amounts of code thus prohibiting rapid transfers between hosts. Finally, limited industry experience and modeling tools for formulating MA solutions to applications in general and IDSs in particular are also factors, as is the additional complexity involved in developing agent-based applications when compared with more traditional forms.

3.2.1. Security

The security concerns related to mobile code are one of the main obstacles to the widespread use of this technology. The MA computing paradigm presents a number of

security threats that are not addressed by conventional security techniques. Standard security techniques must be modified or new techniques invented to address these threats. The security threats can be classified into four broad categories: agent-to-agent, agent-to-platform, platform-to-agent, and other-to-agent platform. The agent-to-agent category represents the set of threats in which agents exploit security weaknesses of other agents or launch attacks against other agents. The agent-to-platform category represents the set of threats in which agents exploit security weaknesses of or launch attacks against an agent platform. The platform-to-agent category represents the set of threats in which platforms compromise the security of agents. The other-to-agent platform category represents the set of threats in which external entities, including agents and agent platforms, threaten the security of an agent platform.

The capability of IDSs to initiate automated responses alleviates the burden placed on system administrators to immediately diagnose suspicious activity and take corrective action. However, this new capability raises security issues within the context of agents. MAs may need to run with administrative or root privileges to perform responses and other tasks. This can cause serious security risk if malicious MAs can be introduced into the system by an attacker. Processes that run with administrative or root privileges introduce new threats and expose the network additional security risks. Moreover, in many instances installing patches or making administrative changes to a system can have an unforeseen effect on existing applications and services.

While these threats exist, they can be mitigated by building upon conventional security techniques. If a MAIDS system can restrict processing to only those agents digitally signed by a security administrator, it greatly reduces the security vulnerabilities, since an attacker can not change the code of an agent to cause it to be malicious. However, an attacker may be able to alter the data of an MA and thereby cause it to perform malicious actions. Additional techniques exist that can be applied to the security problem. Such techniques include mechanisms to control access to computational resources, cryptographic methods to encipher information exchanges, cryptographic methods to identify and authenticate users, agents, and platforms, and mechanisms to audit security relevant events occurring at the agent platform.

More recently developed techniques aimed at mobile code and mobile agent security have for the most part evolved along traditional lines [JANS99]. Techniques devised for protecting the agent platform include Software-Based Fault Isolation, Safe Code Interpretation, State Appraisal, Path Histories, and Proof Carrying Code. Some general-purpose techniques for protecting an agent, include Partial Result Encapsulation, Mutual Itinerary Recording, Itinerary Recording with Replication and Voting, Execution Tracing, Environmental Key Generation, Computing with Encrypted Functions, and Obfuscated Code (Time Limited Black Box). Present day agent systems often include one or more of these mechanisms.

3.2.2. Performance

One of the most challenging problems facing IDSs is improving the speed with which they can identify malicious activity. Not only must IDSs detect attacks quickly, but they must also process system events in real time. This task is becoming ever more difficult as network bandwidth increases. Mobile agent software will generally hinder rather than help an IDS's ability to rapidly process events and detect attacks. MA runtime environments slow down MAIDS especially if they are implemented in slow interpreted languages. The solution is to use MAIDS for some functions, but have core IDS work performed by statically located systems. The performance demands are so high for IDSs that some are starting to be implemented directly in hardware. It may be useful to have MAIDS communicate directly to the more efficient non-MAIDS components.

The performance limitations of scripting and interpreted languages when compared to native code do not offer a promising solution to this problem, as the advantage of heterogeneity offered by these languages come at the cost of speed. When performance criteria are taken into consideration, it is more likely that IDSs will be built using a combination of mobile agents, static agents, and other technologies.

3.2.3. Code Size

IDSs are complex pieces of software. Agents that perform IDS services may thus be required to contain a large amount of code. If these agents are supposed to do operating system specific tasks on multiple operating systems then this code base may get extremely large. The size of MA code may limit the functionality of MAIDS because it will take a long time to transfer an agent between hosts. In addition, such a transfer will require greater computing and network resources. A possible solution to this problem is to have some agents statically locate themselves and have them offer standard Application Programming Interfaces (APIs) to agents that move between machines. Thus, the majority of the code base for the IDS remains stationary while the minority part is mobile.

3.2.4. Lack of A Priori Knowledge

Large enterprise networks are comprised of several different hardware platforms, running several different operating systems, each having different configurations and running different applications. It is not trivial for the mobile agents to have a priori knowledge about how a system is configured, how data is arranged, and still remain lightweight. Static and less transient agents may be more familiar with how data is locally arranged and accessed, and able to act as intermediaries between mobile agents and other platforms. Localized data may be more efficiently manipulated through standard APIs.

3.2.5. Limited Exposure

The client-server computing paradigm is well understood and quite mature as a technology, but the area of distributed control of mobile agent systems is still the subject

of many research efforts. An agent's envisioned autonomous behavior, involving collaboration with other agents at various network locations, creates a dynamic environment that requires new design methodologies and modeling tools to properly formulate and construct agent-based systems. The lack of mature agent design methodologies and modeling tools makes this task difficult, but the problem is likely to be overcome as commercial demand for these product increases and is eventually satisfied.

3.2.6. Coding and Deployment Difficulties

MAs that are developed in-house or purchased from trusted vendors are likely to undergo the same software engineering methods as their non-mobile counterparts in order to assure the quality of code. This standard development process historically produces code with many faults. MAs' inherent capabilities, such as moving and cloning, add more complexity to the design and development process. Given this added complexity, MAIDS will be even more prone to faults than their non-MA counterparts. Further hampering near term MAIDS deployment is a lack of MA design, development, and management tools, needed before any large-scale deployment of agent-based applications becomes feasible. Agent developers and administrators could also benefit from better resource control mechanisms in MA platforms.

4. Innovations in Intrusion Detection Systems

Intrusion detection systems are less than perfect. [MART99] outlines a number of shortcomings of currently deployed IDSs, which are summarized as follows:

- No generic building methodology,
- Lack of efficiency,
- Lack of portability among monitored environments,
- Limited flexibility (includes tailorability, scalability, and dynamic reconfigurablity),
- Limited upgradability of detection techniques,
- Difficult maintenance of rule sets,
- No performance and coverage benchmarks, and
- No good way to test effectiveness.

Developers continue to solve some of these shortcomings through the refinement of existing techniques, but some shortcomings are inherent in the way IDSs are constructed. While mobile agents can help improve IDSs in many areas, they offer no help in others. For example, the ability of an IDS to detect attacks from a single vantage point, by looking at information from a single host, a single application, or a single network interface (i.e., single point detection), is the primary problem facing IDS manufacturers. Mobile agent technology cannot enhance the ability of an IDS to perform single point detection of attacks or reduce false positive rates. Moreover, in most cases, mobile agent technology slows down the ability of an IDS to process events thereby actually decreasing its detection ability. This is a severe limitation for single point IDSs attempting to evaluate events in real time.

This does not mean that MAs are not useful to IDSs. MAs can solve several major problems with IDSs, but more importantly, as discussed below, they can provide IDSs with performance benefits and heretofore unseen capabilities. For example, the mobility of agents make them ideal for detection schemes that follow a “cop on the beat,” “immune system,” or other model.

4.1. Useful Characteristics of MAs

MAs have many characteristics that enable them to enhance intrusion detection technology. Mobility is obviously one of the most important capabilities, and we can certainly benefit from it. However, other agent capabilities also lend themselves to intrusion detection technology. Agent technology and agent applications mimic collections of autonomous and intelligent individuals. Classes of individuals have special purposes and each can operate independently from the others. Each individual talks to other individuals that it meets and exchanges information. This paradigm is in sharp contrast to the traditional programming paradigm where a master logic unit controls a set of slave units. The slaves have no autonomy and perform exactly what the master logic unit commands. Variations of the traditional approach include multiple units with set duties and set communication channels. There may not be a central controller and each

unit may rely on the other units to perform their job. If one unit ceases to function, the other units are not intelligent enough (or don't have the authority) to solve the problem. This traditional distributed programming paradigm works well when components can be relied upon to function. Even by using redundant components, an attacker can disable a small finite number of backups. This traditional design is easy to implement and is an efficient solution to many problems. Agent technology is a great contrast to this design since it attempts to give each agent an understanding of its environment along with the authority to independently make decisions.

MAs are by nature autonomous, collaborative, self-organizing, and mobile. These features are not found in traditional distributed programs, and enable IDSs to implement completely new approaches for doing intrusion detection, some of which are based on analogies found in nature and in society.

Picture a collection of MAs as a colony of bees. Each bee has the ability to fly to flowers and glean pollen just like MAs can move among hosts and process data. Bees have no need to carry large flowers home. Similarly, MAs can avoid having to transfer intrusion detection data located at a collection point, back to a central repository. This research is described under Multi-Point Detection.

Another analogy is to picture a collection of MAs doing IDS work as a colony of ants. Step on one or even hundreds of the workers, and the colony continues to function. In addition, ants can move away when they see a foot descending. MAs can be built with similar attack resistance since they are autonomous and mobile, and killing a few should not harm an ideally constructed MA application. This research is described under the section on Eliminating Single Points of Failure.

MAs can also be viewed as a collection of guards. Security companies do not want to incur the expense of posting a guard in every hall of a building. Instead, they have a guard walk through each hall periodically checking for intrusions. Likewise, MAs enable one to periodically check hosts for security problems without having to install checking software on every host. This research is described under Roaming Agents.

These are just a few examples of how colonies of autonomous mobile agents can benefit intrusion detection technology. Mobility is an important aspect, but that alone is not sufficient. MAs need to be able to operate autonomously and operate in consort with other agents. These features enable new intrusion detection paradigms.

4.2. Research Areas

From the previous discussions, it should be clear that MAs do not add fundamentally new capabilities to actually detecting (non-distributed) attacks or increasing the speed with which one detects these attacks. There is, however, the potential to apply the advantages that mobile agents bring with them to significantly improve the way in which IDSs are designed, constructed, deployed, and operated.

4.2.1. Multi-point Detection[2]

IDSs perform multi-point detection by analyzing events at multiple locations in order to detect distributed or staged attacks. The events may come from multiple hosts, applications, or network interfaces. Multi-point detection is technically difficult even when an IDS can process all of the distributed events. However, few IDSs can afford the network bandwidth required to move the (usually massive) logs of the distributed events to a centralized location for processing. A common solution used by many IDS vendors is to perform filtering and data abstraction on each distributed log file before consolidating the events. However, this further complicates the already difficult problem of detecting distributed attacks because we are now using abstracted data.

Multi-point detection is especially useful in detecting attacks on a network, as opposed to attacks on a host. That is, the objective of a coordinated attack may be to gain access to network resources, not access to a particular host. Mobile agents may learn of this strategy by correlating attacks on gateways, hosts, modems, servers and other points of exposure to outside entities. Host-based systems will only realize that individual components are under attack, and cannot speculate on the larger strategy.

MAs may benefit multi-point detection technology by allowing the massive amount of distributed log data to stay stationary. An MA analysis engine could move among data pools to perform multi-point detection on the original logs. This is an ideal application of MAs, where one needs to transfer the computation to the data instead of the data to the computation. Research is needed to determine how an agent can efficiently collect and analyze data in a piecemeal fashion as it travels around to different information sources to perform multi-point analysis. That is, instead of the information moving one time to a central site and being analyzed in a collective fashion, it would now need to be reduced and moved multiple times (i.e., with each hop) and done so efficiently.

4.2.2. Attack Resistant Architectures

IDSs often use hierarchical architectures for reasons of efficiency and centralized control (for more information see Appendix B on architectural issues). Envision an IDS that implements a hierarchical architecture without the capability to dynamically reconfigure relationships to compensate for failure of key components. Single point detection occurs at the leaves (as well as data gathering for multi-point detection). The results from the leaf nodes are sent up the hierarchy to internal nodes that perform data abstraction (and possibly multi-point detection). The data is continuously abstracted until it reaches a command and control node at the root. This design has no redundant communication lines, resulting in many single points of failure. An attacker can cut off a control branch of the IDS by attacking an internal node or decapitate the IDS by taking out the root.

[2] While many IDSs are starting to perform multi-point detection, there is little evidence that such attacks exist except in limited forms. Scanning and worms are the most obvious kinds of attacks that are best detected by gathering data from multiple locations in the network. However, we have not been able to identify many attacks whose detection benefits from multi-point detection.

Even a small number of redundant backup hosts created for each key node is not beyond the reach of a knowledgeable and determined attacker.

While several solutions to this problem exist, MAs readily apply since MAs are by nature autonomous. A collaborative multi-agent system can be self organizing and thus adaptive to attack. Some obvious areas for research and experimentation include:

- Completely distributed and decentralized IDS architectures where there exist no single points of failure and there exist numerous redundant information pathways.
- Standard hierarchical IDSs where an MA backs each node up and restores any lost functionality out of sight of an attacker.
- Mobile agent IDSs that relocate when any suspicious activity is detected.

4.2.3. Abstract Interfaces

MAs are autonomous and usually small in order to enhance transmission efficiency. Thus, a single MA is unlikely to perform all event generation tasks (logging), data abstraction tasks, and attack detection tasks. Thus, multiple autonomous MAs must communicate with one another about events and attacks. Also, MAs may need to communicate with statically located traditional event generators. These communication requirements may be a great hurdle in developing MAIDS.

We need to define abstract interfaces for event generation, abstracted events, and attack descriptions. Several alternative forms of abstraction include representation as a management information base, a program library, or an agent dialogue. The information must be specific enough for each MA to perform their desired computation (e.g. detect an attack), but general enough to avoid being computationally expensive. A flexible scheme is needed whereby MAs can subscribe for the specific information that they need. If too many MAs request information and the system becomes overloaded, an elegant back off algorithm is needed to strip excess requests.

4.2.4. Knowledge Sharing

Often, several completely independent IDSs are deployed in an organization. Ideally these IDSs would share information about recent attacks in a way that would enhance their ability to detect future attacks. This is not specifically a MAIDS research field, but MAs enable new paradigms of knowledge sharing by easily implementing mobile and autonomous components. While we can build distributed and decentralized IDSs using MAs, it is unclear how the MAs can ideally share knowledge about events in the network. Knowledge sharing architectures need to be developed that take advantage of the autonomous and distributed nature of MAs.

A more short-term view of research with knowledge sharing and MAs is to attempt to overcome the shortcomings of network based IDSs using MAs. Network-based IDSs are typically located next to an organization’s firewall and sniff the traffic of possibly thousands of hosts. In this situation, it is impractical for an IDS to emulate the protocol

stack of each host that it is protecting. In most cases, the IDS does not know what operating systems are running on each host. Thus, IDSs are forced to filter network packets using a generic network protocol stack. Attackers take advantage of this by sending a target host packets that are interpreted differently by the IDS and by the target. This is done using various fragmentation, sequence number, and packet flag tricks [PTAC98]. The attacker then penetrates the target while the IDS is blind to the attack.

MAs can assist IDSs with this problem by coordinating between network sniffing IDSs and target hosts. If the network-based IDS is able to emulate multiple protocol stacks, then MAs may probe hosts to find out what operating system and applications are running. Alternatively, MAs can be deployed to hosts to detect attacks that utilize protocol stack differences. Thus, the MAs can work as host-based IDSs in conjunction with a network-based IDS in order to catch attackers trying to deceive the network-based IDS. It is possible that deploying the host-based MAIDS may be computationally costly and so this may only be done when somewhat suspicious traffic patterns are seen that are not themselves identifiable as an attack. There are many variations, but in all MAs can work with statically placed network based IDSs to prevent/detect an attacker's attempts to disguise himself.

4.2.5. Roaming Agents

Each agent may perform specific tests (much like a mobile sensor) and randomly roam the network. When the test indicates some possibility of an intrusion, the agent may ask for additional tests at the site. Only after the suspicion level has been raised high enough is the actual alarm given. Notice that the attack is confirmed by executing only relevant tests, and the entire suite of tests does not have to remain resident at every node.

Random sampling has been successfully used for many years for quality control in manufacturing. Fundamentally, if a random sample detects a problem, then a more comprehensive series of tests needs to be run. The mathematics is well understood and the parameters may be calculated. For example, it is possible to calculate the average length of time before an attack is discovered, the average number of systems likely to be infected before discovery, the bandwidth consumed, and the average computation at each node. The mobile agents themselves can also have statistical properties such as the rate at which they test nodes, their size, the diversity of nodes visited, etc. Changes to these statistical features may indicate an attack on the agent itself.

Since mobile agents roam throughout the network, they may not be constantly resident at every node. Consequently, those nodes without a resident agent are vulnerable until an appropriate agent arrives. This situation, however, is not as deleterious as it first appears. Very high "false positive" alarm rates plague conventional IDSs. Consequently, administrators ignore most alarms and rarely detect intrusions during their first appearance. Rather, clues gradually accumulate until the administrator makes a determined effort to explain some anomaly. It is therefore not necessarily true that an MA random sampling structure will significantly degrade the detection process; the results

depend upon the attacks and the criticality of the system. In addition, such a system may work well in conjunction with a traditional IDS.

Roaming MAs can be used as detectors in an anomaly based IDS. Roving MA detectors can generate events from locations in the network previously inaccessible to statically installed anomaly detectors. Agents collect statistics about the performance of the network or hosts, which is not meaningful at individual nodes, but may be meaningful in aggregate. The data should exhibit some regularity, and significant changes would be cause for concern. By correlating events from across the network, we may be able to detect new attacks and with a lower false-alarm rate.

4.2.6. Unpredictability

An attacker may successfully break in to a host with a conventional host based IDS and not be detected immediately. This could happen either because the attack was too clever for the IDSs or because the IDSs only scans the host for attacks periodically due to performance considerations. In this event, the attacker now has free reign to inspect and alter the IDS, install backdoors, and remove evidence of the attack from the audit log.

A MAIDS solution is also vulnerable to the same kind of attack. However, MAs offer some benefits for detecting such tampering. As each new mobile agent arrives at the host, it embodies a fresh copy of the intrusion detection procedures. Some of these checks may make sure that the agent platform is unaltered. For example, an agent could calculate a checksum of a static system file or perform a similar integrity check on some aspect of the platform, and report the result upon return to a point where validity can be determined. An unexpected result would warrant remedial action.

The inter-arrival times of agents, their reporting mechanism, and the exact nature of their detection algorithm may also be made unpredictable. Depending on the goal of the attack, the attacker may not be willing to kill the agent platform, as it would be suspicious for a host to suddenly not accept MAIDS. Instead, the attacker can modify the MA platform to blind incoming agents. The degree of difficulty in modifying the agent platform to blind, but not otherwise hamper agents could vary widely. However, if standard IDS interfaces exist the effort could be as simple as replacing the appropriate system library. One way of dealing with the problem of platform subversion is to incorporate the agent platform into the kernel. Even though the kernel can be subverted, the difficulty of modifying the kernel raises the level of difficulty of disabling the IDS. Hardware solutions exist for preventing tampering with the host platforms, but they tend to be prohibitively expensive for general use.

The areas for research revolve mainly around applying unpredictability as a complement to other mechanisms. While unpredictability does not apply to denial of service or other short lived attacks, it does address those attacks that attempt to use the penetrated host as a base of operation for further exploits and leave potentially incriminating evidence. The goal is to increase the likelihood that such a successful penetration attack does not elude detection indefinitely.

4.2.7. Genetic Diversity

IDSs created with MAs can be viewed as a collection of autonomous entities. However, usually each MA will not be unique in the sense of having a distinct set of instructions from which it runs. Typically, one will create classes of agents and members of the same class will have different data, but the same instructions. For example, one class of agents may run around the network looking for a particular vulnerability.

The problem with this approach, and with standard IDSs, is that the tests that are run are predictable. With standard IDSs, an attacker can buy a copy of the IDS and figure out the exact signature used to detect the attack. This knowledge gives the attacker an advantage when penetrating a network.

What if agents that detected a particular attack each had a slightly different detection signature. The attacker then could not predict exactly what detection algorithm is going to be used to detect his attack. The drawback is that by using different detection signatures, each will have a different false positive and false negative rate. We are forcing ourselves to use non-optimal signatures. However, these rates can be bound to a reasonable range and then we can still take advantage of using MAs that detect particular attack differently.

One way to teach agents different ways to detect attacks is to give agent base knowledge about an attack and have them automatically learn their own technique for detecting it. Have a human describe, in some standard language, every aspect of an attack. Create agents that can read the description language. Using a variety of machine learning algorithms, agents can use different aspects of an attack and formulate an attack signature. If one puts these agents in an isolated network and launches the attack, through a feedback loop the agents can calculate their detection rates and false positive rates. Agents with low detection rates can make small modifications to their randomly generated attack signature in an attempt to improve. The result will be a large set of agents each of whom detects an attack with a different signature. The better agents can be deployed throughout a network. Thus, MAs can automatically learn attack signatures and learn different signatures. This will prevent the attacker from predicting the exact signatures used and thus enhance the distributed IDS.

5. Innovations in Intrusion Response

As their name implies, IDSs have traditionally focused on detecting attacks. While detection serves a useful purpose, oftentimes a human does not analyze the reports from an IDS for some time. This gives an attacker a window of opportunity in which to freely operate before being countered by a systems administrator. During this time, the attacker may steal critical data, install invisible back doors, use the penetrated host to attack other sites, or subtly corrupt information. Ideally, an attacker should not be given any time to extend their grasp on a network.

Because of this, many IDSs are beginning to implement automated response capabilities. The IDS detects an attack and immediately responds in order to kick the attacker out of the network. This sounds simple, but in practice it is very difficult to accomplish. For safety and technical concerns (i.e., the response could be used to an attackers benefit), IDSs initiate only very weak responses. We present ways in which MAs can help solve this problem, however, first we explore existing automated response mechanisms and the challenges facing them.

5.1. Existing Response Mechanisms

Existing response mechanisms in commercial IDSs are somewhat weak when compared to the goal of automatically ousting an attacker from the network. The automated response mechanisms currently implemented fall into two categories: enhanced notification and attacker filtering.

Enhanced notification mechanisms are designed to inform systems administrators of important attacks as soon as possible. They thus decrease the time window an attacker has before being countered by a human. In the event of an important attack, IDSs e-mail system administrators, pop up notification windows on their monitors, or even page them.

Attack filtering methods actively stop an attacker. One popular technique is to interrupt a TCP connection between an attacker and a target. IDSs do this by sniffing the malicious connection to determine packet sequence numbers and then by inserting reset packets to kill the connection. Another popular attack filtering technique is to dynamically change the routing permissions table in routers and firewalls. Typically, an IDS disallows packets from an attacker's IP address or subnet from traversing the network's routers or firewall. The target can also be cut off using this technique in order to prevent the attacker from launching attacks from the target host. Using these attack filtering methods, an IDS attempts to disrupt the attacker's access to the target and stop the attacker from sending more malicious packets into the network.

While IDSs are making great strides in the realm of automated responses, the current functionality is not sufficient. The existing response mechanisms are too weak to stop sophisticated attackers. Currently implemented response mechanisms are not sufficient because they assume that attacks take time to launch and that attackers are limited to

using a single IP address or subnet. However, modern computer attacks are usually launched using automated attack programs. These programs break into computers very quickly using only a few packets and can penetrate a host before an IDS detects and responds to the attack. The attack program can quickly install a back door. Then the attacker approaches the compromised machine from a new IP address, uses the back door, and the IDS does not detect the apparent normal entry into the host. Another problem occurs when the attacker is launching denial of service attacks. In this case each packet may be spoofed with a different IP address thus making filtering of the attack packets impossible.

5.2. Ideal Response Mechanisms

Assume an attacker has penetrated a network and corrupted some hosts. The ideal response gathers evidence of the attacker's activity, removes the attacker's access to the network, undoes the damage, and reconfigures the network to resist the attacker's penetration technique. It is impossible in today's environment to automate this ideal response since humans themselves have great difficulty enacting it. We can not automate what we ourselves can not do. However, we can automate an approximation of this ideal response. Our approximation should have the following capabilities not found in current automated response software:

- *The ability to dynamically modify or shut down the target.* This capability enables the IDS to automatically remove the intruder from the target, protect it from further damage by shutting it down, or perform an enhanced audit of the attacker's actions.

- *The ability to dynamically modify or shut down the attacking host.* With insider attacks, this enables the IDS to automatically stop the generation of the attack as well as record evidence of the attacker's actions.

- *The ability to determine the host which is launching the attack.* When attack packets are being spoofed, they can only be traced to an Ethernet wire by querying each router as to the source of the packets. Once the correct Ethernet is found, each host on the Ethernet must be analyzed in order to determine which is responsible for launching the attack. Thus an IDS must provide the capability to trace the path of an attacker.

- *The ability to monitor all network traffic to and from the target.* It is necessary to record for evidence the packets that the attacker sends to the target. In addition, it is necessary to record packets leaving the target since it may be used as a jump point to penetrate other hosts

- *The ability to modify the routing and firewall permission tables on every firewall and router*. We often want to isolate the attacker or target in order to prevent further damage. Such isolation can limit legitimate traffic and so we want to optimally place the filters such that the attacker is constrained the most while allowing the most legitimate traffic.

This list of required capabilities needed to approximate an ideal response implies that security services are installed on every host and network device. This does not mean, however, that every network component needs to have an IDS installed that is aware of the rest of the network. Nor does it imply that these security services need to be proprietary solutions. Each network component could have a security server installed that performs some detect and response functions. The security servers may have a standard API that would allow them to be used by a distributed IDS scheme. Thus, every network device needs detect and respond capability but no single proprietary scheme needs to be installed everywhere. This vision is being realized in the router market where one can now dynamically change routing filters from remote locations.

It is unlikely that all network devices will soon be installed with security servers that can be controlled with a common API. It would be very difficult to get the security community to agree on such a standard and splintering standards promote proprietary solutions. However, we also think it is also unlikely that companies will want to install a proprietary security solution on every host and network device. The cost of installation and maintenance of such a scheme would be overwhelming. Therefore, it seems unlikely that, without MAs, the security infrastructure that is needed to approximate the ideal response can be cost effectively implemented.

5.3. An MA Automated Response Solution

Mobile agent technology can solve the problem of installing and maintaining the security infrastructure needed to approximate an ideal response. Using MAs, it is not necessary to install a security server on every device as the MAs can automatically travel through the network and install the appropriate software on the appropriate types of network devices. This way, companies are not locked into using a single proprietary solution and uninstalling one solution and installing another can be almost automatic. MAs can provide the security servers that are required at each network device and also implement the distributed IDS that detects attacks and issues responses.

MAs enhance a system's ability to automatically respond because MAs make it possible to make all network components part of the same security scheme. Responses can be initiated at any place in the network, which gives systems the capability to optimize the locations at which they initiate responses. Furthermore, MAs enhance an IDS's ability to trace an attacker through the attacked network, to respond on the target, respond on the attacker, and to collect network/host evidence about the attack.

5.4. Research Areas

Like many other areas, MAs do not add fundamentally new capabilities to automated response. However, MAs may help transition some automated response ideas from an impractical and costly scheme into a solution that can be implemented in a cost-effective manner. Thus, many of these research areas are not specifically about MAs, but they are

areas that need MAs to be practical and they are areas that have been ignored previously because of their seeming impracticality.

5.4.1. Automated Tracing of Attackers

We envision future IDSs tracing attacker's paths within IDS enabled networks. This feature is useful for two reasons. First, attackers often log into a chain of many hosts before attacking a target. Thus, to find the attacker one must trace back along the chain. Second, attackers sometimes can spoof their source address. It is necessary to determine the actual source of these packets and the way to do this is to trace the packets from LAN to LAN until the source LAN is found. Finding the actual host launching the packets requires visiting the hosts within a LAN since attackers can spoof MAC addresses as well as IP addresses. Many attacks originate from outside of a network and thus could not ultimately be traced, however insider attacks are a large concern. Whether an attack is from an insider or an outsider, IDSs need to determine this and locate the location of the attacker as best as possible.

Effectively tracing an attacker from LAN to LAN requires the ability to sniff on every Ethernet segment in the network. Finding the host on an Ethernet segment that is lying about its identity requires analyzing each host on an Ethernet segment. Thus, to adequately trace an attacker through a network an IDS needs the capability to sniff on every Ethernet segment and to analyze every host. Ordinarily, the infrastructure required to support this kind of tracing would be prohibitively expensive. However, MAs provide a very cheap way to do this provided that MA platforms are widely installed.

There are several issues that a tracing system should address. If an attacker has compromised a host, then the MA platform may not function. Thus, any tracing system must gather data from many locations and compare the results. In addition, not all hosts may support MAs and thus the system will need to understand the topology of the network and sniff traffic at the optimal locations in order to pick up the attacker's trail. Finally, what the attacker sends along each link in his attack chain may be completely different data. For example, the attacker could use link encryption. In this scenario, to trace the attacker the system may need to apply sophisticated artificial intelligence techniques or decision-making algorithms.

Assuming the existence of MA platforms, the tracing system can automatically deploy across the network with very little installation time needed. The main research problem in building these systems is to anticipate a clever attacker. An automated tracing system needs to:

- Intelligently collect evidence of an attackers trail,
- Adaptively operate when some hosts are not participating in the tracing scheme (e.g. not running an MA platform), and
- Have the ability to search many paths simultaneously while not overwhelming the network with agents.

5.4.2. Automated Evidence Gathering

Currently, it is impractical to automatically gather evidence for an attack from many different sources. The problem is having the right software running at the right place at the right time. MAs offer the ability to run anything, anywhere, at any time. It is therefore conceivable that evidence may be gathered from different hardware platforms, different operating systems, and even different applications such as web servers.

It is very easy to audit so much information on a host that the capacity to store the logs fills up quickly. It is even easier to run up against this barrier when saving the audit logs of every host on the network and trying to record the network traffic on each Ethernet wire. Thus, systems administrators can not record all evidence in a network.

MAs may determine what types of evidence are most needed for that network, for the type of attack being investigated, and at what locations in the network. This is likely to be different if the evidence is for internal use as compared to legal use. The MAs may then move to the appropriate locations and start up relevant audit services (if they are not already enabled). MAs can thus intelligently audit the network by dynamically reconfiguring the audit capabilities of every host. The MAs can strongly audit suspicious or important network locations while lightly auditing other areas. Thus, MAs can reduce the amount of wasted evidence that must be stored.

Evidence gathering MAs should be in constant contact with IDSs that provide it with suspicious locations in the network. With this collaborative technique, evidence gathering MAs will often be able to fully audit host and network traffic generated by attackers.

5.4.3. MA Operations on an Attacker's Host

In the event of an attack, automated responses normally occur in the network at routers or firewalls. These elements typically try to separate the attacker from the target. However, if possible it would also be beneficial to launch automated responses on the attacker's host. Such a counter-attack may not succeed as the attacker has control of his own host and so this technique would not replace router or firewall based responses. However, responding on the attacker's host gives an IDS a much greater power to restrict the attacker's actions. Without MAs, it is unlikely that an IDS could get enough access to an attacker's host in order to initiate responses. Because of this, the field of responding to an attacker on his own host has not been researched. Having MA platforms installed throughout a network will enable IDSs to initiate these kinds of responses and thereby necessitate this type of research.

There are not many responses that can be taken by routers and firewalls to stop attacks. The responses consist of filtering out certain types of traffic or killing connections. Alternatively, there are an enormous number of useful responses that could be implemented on an attacker's host. We need to research what types of responses are most useful. We need to research what types of responses work best against an attacker using particular types of attack tools.

5.4.4. MA Operations on a Target Host

When attacked, it is obviously vital to automatically respond on the target host. Such a response can prevent the attacker from using the penetrated host to further compromise one's network and to recover damage done by the attacker.

Much of the research defined for MA operations on the attacking host carry over to this area. However, the attacker will not typically be as firmly entrenched on the target host as he is on the attacking host. The attacker may only have user access, not administrator access to the host, and the attacker probably has not subverted many of the standard host services. Thus, if an IDS acts quickly, the target host will be a correctly functioning system that has an attacker as one of its users.

An MA can be sent by an IDS to the target to engage the attacker. Since the MA is automated it may often move faster than an attacker's manual actions. However, an attacker may use automated tools that can act faster than the MA. Despite the fact that the MA may lose the battle for the target, an IDS should dispatch an MA in the hopes of its success.

The MA needs to determine how the attacker is controlling the system. As the attacker attempts to gain privilege on the system, the MA should counter these attempts and attempt to remove the attacker. The MA will probably need to make use of the attack detected by the IDS in order to figure out how the attacker is controlling the system. Depending upon the type of host penetrated, the MA may simply take the host down or may try and keep vital services operational while delaying the attacker until a systems administrator can evaluate the situation.

5.4.5. Isolating the Attacker/Isolating the Target

As appealing as they may sound, actions to automatically respond on the target and attacking host may fail. It is vital that uncompromised machines respond at the network level to limit an attacker's actions. Three strategies exist to deter an attacker: isolating the target, isolating the attacker, and creating a cut set between the target and attacker. The fact that MAs can travel to all network elements to implement responses is what enables them to perform these strategies.

Isolating the target involves encircling the target with a barrier such that the target can not launch attacks on the rest of the network. A simple way to erect such a barrier is to cut off all network traffic along the edge of the circle. More sophisticated filters can be researched that allow "safe" services through the circle. The research problem is that the barrier will probably cut off some legitimate communication. We must take the network topology into account and encircle the target such that as much vital communication as possible is allowed to continue while still encircling the target. Furthermore, some systems that we can not allow to be compromised must be placed outside of the circle while other systems that must communicate with the target must be placed inside of the

circle. This is an optimization problem that will also require research to properly implement.

Isolating the attacker involves the same logic as isolating the target. One wants to draw a circle around the attacker, as tight as possible, such that the attacker can not launch attacks through the circle. Again, some hosts may need to communicate with the attacker and thus must reside within the circle.

A third option is to create a *cut set* between the target and the attacker. This merely separates the two protagonists with an impenetrable boundary. It does not attempt to stop that attacker or the target from attacking other hosts in the network. Creating a cut set is useful when one is primarily concerned about the attacker reaching the target. The cut set technique always can be set up to eliminate no more legitimate communication than the isolating attacker or isolating target techniques.

5.4.6. MA Operations on Attacker and Target Subnet

We advocated using MAs to respond on attacker and target hosts because of the flexibility the MAs would have in initiating responses. However, the drawback was that these MAs would sometimes fail in the face of sophisticated attackers. We advocated isolating attackers and targets at routers and firewalls so that the attacker could not thwart the response, but this means that sometimes legitimate traffic can be cut off. A third response alternative that can be implemented using MAs is for an MA to travel to an uncompromised host on the attacker's or target's subnet. From this position, the MA can launch attacks against the attacking or target host to stop it from functioning. MAs may utilize well-known vulnerabilities or may simply flood the attacker and target hosts with packets. Since MAs can travel to multiple hosts, they can muster sufficient network resources needed to flood any attacked machine. The MAs can then keep the target and attacker from doing further damage until a systems administrator evaluates the situation.

6. Summary

At first glance, mobile agent technology offers much to the field of intrusion detection. The idea of mobile and autonomous components intuitively seems useful in intrusion detection and many other applications. However, it is difficult to realize the benefits of mobile agent technology in practice. Despite these difficulties, the technology appears to provide valuable extensions to current capabilities. Although the barriers to creating practical mobile agent systems are high, the ability to move a running program from one hardware platform to another is a useful feature. Ultimately, as the security, performance, emerging technology, and standards barriers that inhibit this technology fall, mobile agents will enter mainstream use.

Not only do mobile agents appear to be useful in general, but they appear useful to IDSs. Mobile agents may enhance the performance of IDSs and even offer them new capabilities. However, obtaining these benefits is not easy and will require a substantial commitment of resources to research.

There are three main research areas for using MAs to do intrusion detection: performance enhancements, IDS design improvements, and response improvements. Within IDS design improvements there are three categories of research: new detection paradigms, new architecture paradigms, and improvements over existing designs. We outline the three areas below and rate the importance of each research area. Each specific research area maps to a research section discussed above. Importance is rated for each area as high, medium, or low. Importance is a subjective[3] measure combining the chance of successful research with the possible impact on the intrusion detection field. It is our hope that these ratings will help guide future research toward the more fruitful areas in using mobile agents to perform intrusion detection.

Performance Enhancements

Research Objective: Design MAIDS that take advantage of mobility and autonomy to obtain better performance than equivalent non-mobile IDSs. Performance can be measured as event analysis speed as well as system up time.

Research Area	Rating
Overcoming network latency	Medium
Reducing network load	Low
Scalability	Medium

Table 1: Performance Enhancement Research Area Ratings

[3] These ratings are extremely subjective. They are merely the best guess of the authors. Our rationale for these ratings is incomplete at best and thus we forgo attempting any justification.

IDS Design Improvements

Research Objective 1: Use MA technology to enable novel paradigms for detecting attacks.

Research Area	Rating
Multi-point detection	High
Roaming agents	High
Genetic diversity	Low

Table 2: Attack Detection Research Area Ratings

Research Objective 2: Use MA technology to enable novel paradigms for IDS architectures.

Research Area	Rating
Attack resistant architectures	High
Abstract interfaces	Low
Unpredictability	Medium
Knowledge sharing	Medium

Table 3: Architecture Research Area Ratings

Research Objective 3: Use MA technology to overcome shortcomings of current IDS architectures.

Research Area	Rating
Asynchronous execution and autonomy	Medium
Structure and composition	High
Adapting dynamically	Medium
Operating in a heterogeneous environments	Low
Robust and fault-tolerant behavior	High

Table 4: Architecture Research Area Ratings

Response Improvements

Research Objective 4: Use MA technology to enable novel and efficient automated responses to attacks.

Research Area	Rating
Responding with MAs:	
• Operations on the attacker's host	Medium
• Operations on the target host	Medium
• Operations on attacker/target networks	High
• Methods of isolating the attacker/ target	High
Automated tracing with MAs	High
Automated evidence gathering with MAs	Medium

Table 5: Response Improvement Research Area Ratings

7. References

[ANDE80] Anderson, James P., "Computer Security Threat Monitoring and Surveillance," Technical Report, James P. Anderson Co., Fort Washington, PA, April 1980.

[AMOR99] Amoroso, Edward, *Intrusion Detection*, Intrusion.net Books, Sparta, New Jersey, 1999.

[ASAK99] M.Asaka, S.Okazawa, A.Taguchi, and S.Goto, "A Method of Tracing Intruders by Use of Mobile Agents," INET'99, June 1999.

[BALA98] Jai Balasubramaniyan, Jose Omar Garcia-Fernandez, David Isacoff, E. H. Spafford, and Diego Zamboni, "An Architecture for Intrusion Detection using Autonomous Agents," Department of Computer Sciences, Purdue University; Coast TR 98-05, 1998. <URL: http://www.cs.purdue.edu/coast/coast-library.html>

[BARR98] Barrett, Michael, W. Booth, M. Conner, D. Dumas, M. Gaughan, S, Jacobs, M. Little, "Intelligent Agents System Requirements and Architecture," Report to ATIRP, p. 5, October 1998.

[BAUE88] Bauer, David S. and Koblentz, Michael E., "NIDX: An Expert System for Real-Time Network Intrusion Detection," Proceedings of the Computer Networking Symposium, pp. 90-106, April 1988, Washington, DC.

[BRAD97] Jeffrey M. Bradshaw, "An Introduction to Software Agents," In Jeffrey M. Bradshaw, editor, *Software Agents*, chapter 1. AAAI Press/The MIT Press, 1997.

[CHES95] Chess, D., B. Grosof, C. Harrison, D. Levine, C. Parris, G. Tsudik, "Itinerant Agents for Mobile Computing," IBM Research Report, RC 20010, March 1995. <URL: http://www.research.ibm.com/massdist>

[CIDF] "Common Intrusion Detection Framework Specification," Version 0.6, CIDF working group. <URL: http://seclab.cs.ucdavis.edu/cidf/ >

[CONN99] Michael Conner, Chirag Patel, Mike Little, "Genetic Algorithm/Artificial Life Evolution of Security Vulnerability Agents," Army Research Laboratory Federal Laboratory 3rd Annual Symposium on Advanced Telecommunications & Information Distribution Research Program (ATIRP), February 1999.

[CROS95] Mark Crosbie and E. H. Spafford, "Active Defense of a Computer System Using Autonomous Agents," Department of Computer Sciences, Purdue University, CSD-TR-95-008, 1995.

[DENN87] Denning, Dorothy E., “An Intrusion Detection Model,” IEEE Transactions on Software Engineering, Vol. SE-13, No. 2, pp. 222-232, February 1987.

[FARM96a] Farmer, W.M., J.D. Guttman, and V. Swarup, “Security for Mobile Agents: Authentication and State Appraisal,” Proceedings of the 4th European Symposium on Research in Computer Security (ESORICS ’96), pp. 118-130, September1996.

[FARM96b] Farmer, W.M., J.D. Guttman, and V. Swarup, “Security for Mobile Agents: Issues and Requirements,” Proceedings: National Information Systems Security Conference, pp. 591-597, October 1996.
<URL: http://csrc.nist.gov/nissc/1996/papers/NISSC96/paper033/>

[FINI94] Finin, T., R. Fritzson, D. McKay, and R. McEntire. “KQML as an Agent Communication Language,” Proceedings of the Third International Conference on Information and Knowledge Management (CIKM ’94), ACM Press, Nov. 1994.

[FRIN98] Frincke, D., Don Tobin, Jesse McConnell, Jamie Marconi, Dean Polla, “A Framework for Cooperative Intrusion Detection,” Proceedings of the 21st National Information Systems Security Conference, pp. 361-373, October 1998. <URL: http://csrc.nist.gov/nissc/1998/papers.html>

[HANS97] Hansoty, Jatin N., “LAVA: Secure Delegation of Mobile Applets,” Master’s Thesis North Carolina State Univ., 1997. <URL: http://shang.csc.ncsu.edu:80/lava.html >

[HARR95] Harrison, C.G., D.M. Chess, A. Kershenbaum, “Mobile Agents: Are they a good idea?,” IBM Research Report, March 1995.

[HEBE90] L.Todd Heberlein, G.V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, D. Wolber., "A Network Security Monitor," Proceedings of the Symposium on Research in Security and Privacy, pp.296-304, May 1990.

[HELM98] Guy Helmer, Johnny S. K. Wong, Vasant Honavar, and Les Miller. “Intelligent Agents for Intrusion Detection.” Proceedings, IEEE Information Technology Conference, Syracuse, NY, pp. 121-124, September 1998.
<URL: http://www.cs.iastate.edu/~ghelmer/ieee-1998.ps>

[BACE99] Bace, Becky, “An Introduction to Intrusion Detection and Assessment for System and Network Security Management,” ICSA, Inc.
<URL: http://www.icsa.net/services/consortia/intrusion/educational_material.shtml>

[JACO99] Jacobs, S., D. Dumas, W. Booth, M. Little, “Security Architecture for Intelligent Agent Based Vulnerability Analysis,” Proceedings: 3rd Annual Fedlab Symposium on Advanced Telecommunications/Information Distribution Research Program, pp. 447-451, February 1999, College Park, MD.

[JANS99] Wayne Jansen and Tom Karygiannis, “Mobile Agent Security,” National Institutes of Standards and Technology, NIST SP 800-19, August 1999.

[JUMPIN] “Jumping Beans Security,” Ad Astra Engineering.
<URL: http://www.jumpingbeans.com/Security.html>

[LANG98] Danny Lange and Mitsuru Oshima, *Programming and Deploying Java Mobile Agents with Aglets*, Addison-Wesley, 1998.

[LEE99] W. Lee, S.J. Stolfo, and K. Mok, “A Data Mining Framework for Building Intrusion Detection Models,” Proceedings of the IEEE Symposium on Security and Privacy, 1999. <URL: http://www.cs.columbia.edu/~sal/JAM/PROJECT/>

[LUNT88] Teresa F. Lunt and R. Jagannathan, “A Prototype Real-Time Intrusion-Detection Expert System,” IEEE Symposium on Security and Privacy, April 1988.

[MARR98] Marreale, P., “Agents on the Move,” IEEE Spectrum, April 1998, pp. 34-41.

[MART99] Stefano Martino, “A Mobile Agent Approach to Intrusion Detection,” Joint Research Centre-Institute for Systems, Informatics and Safety, Italy, June 1999.

[MILL99] Kevin Mills et al., “Wireless Information Technology for the 21st Century,” Draft White Paper, February 1999.

[NEUM94] B. Clifford Neuman and Theodore Ts'o. “Kerberos: An Authentication Service for Computer Networks, ” IEEE Communications, 32 (9), pp. 33-38, September 1994. <URL http://nii.isi.edu/publications/kerberos-neuman-tso.html >

[PTAC98] Thomas H. Ptacek and Timothy N. Newsham., “Insertion, Evasion, And Denial Of Service: Eluding Network Intrusion Detection,” Technical Report, Secure Networks, Inc., January 1998.
<URL: http://www.nai.com/services/support/whitepapers/security/IDSpaper.pdf >

[SEBR88] Michael M. Sebring et al., “Expert Systems in Intrusion Detection: A Case Study,” Proceedings, 11th National Computer Security Conference, pp. 74-81, October 1988.

[SMAH88] Stephen E. Smaha, “Haystack: An Intrusion Detection System,” Fourth Aerospace Computer Security Applications Conference, Orlando Florida, pp. 37-44, December 1988.

[SMIT88] Jonathan Smith, “A Survey of Process Migration Mechanisms,” Operating Systems Review, 22(3), ACM Special Interest Group on Operating Systems, pp. 28-40, July 1988.

[TENN97] David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden, “A Survey of Active Network Research,” IEEE Communications Magazine, Vol. 35, No. 1, pp.80-86, January 1997.

[WHIT96] Gregory B. White, Eric A. Fisch, and Udo W. Pooch, “Cooperating Security Managers: A peer-based intrusion detection system,” IEEE Network, 10(1), pp.20-23, January/February 1996.

[WU96] Wu, S.F., M. S. Davis, J. N. Hansoty, J. J. Yuill, S. Farthing, J. S. Webster, X. Hu. “LAVA: Secure Delegation of Mobile Applets,” Technical Report 96/42, Center for Advanced Computing and Communication, North Carolina State Univ., Raleigh, NC, October 1996.

Appendix A - IDS Operational Environments

This appendix provides a brief overview of the operational environment in which we foresee the IDS being deployed. An effort is made to take new and emerging technologies, such as Active Networks and Virtual Private Networks (VPN), into consideration in order to forecast what networks in the next 5-10 years might look like and what new challenges may arise. The operational environments for IDSs will have the following characteristics:

- Network users will continue using combinations of Intranets, the Internet, LANs, and WANs, to meet their computing needs and these networks will feature much higher bandwidths, support more applications, rely on intermediate nodes that use more open source code, and have more diverse network entry points such as cable modems and wireless devices.

- The use of Virtual Private Networks (VPN) will increase as enterprises begin using both public and private network segments for their networking needs. VPNs will make it more difficult for network-based IDSs to gather enough information from the encrypted network traffic and the IDSs will need to cooperate with host-based techniques to detect and diagnose suspicious activity.

- Networks will be increasingly segmented within the enterprise, with firewalls or other technologies limiting access to these network segments, with each segment having different security policies, and being under different administration domains. MAs operating within this environment will have different privileges within each domain and they must be able to negotiate several security policy management issues.

- In order to keep up with the increasing bandwidth of networks, parts of IDSs will be placed on routers and also be implemented in hardware [AMOR99]. Some attack signatures are stateless while others require state. The stateful attack signatures consume the majority of computing resources. Firewall vendors have begun including stateless IDSs on firewalls and IDSs can now communicate with routers using the MIB interface so that the IDS can initiate packet filtering responses.

- More and more people around the world will be accessing public networks. This will result in more sources of attack in which a number of individuals and organizations can pool distributed resources to attack individual targets. Moreover, automated network attack toolkits will become easier to use and require less sophistication to launch.

- System administrators will need to rely on more COTS host-based, network-based, forensic, and data reduction tools from various vendors. These tools will need to be interoperable.

- Mobile users will be accessing public and private network more often and through a variety of devices, including notebook computers and Personal Digital Assistants (PDAs). Wireless devices will allow users and devices to more easily access public and private networks. The desktop PC will be replaced, or augmented, with clusters of wireless devices. The simplest such clusters will simply replace all the wires connecting PCs, monitors, mice, keyboards, speakers, and printers. While these devices need not be connected directly to the Internet, devices such as speakers may be able to directly tune in radio broadcasts from the Internet while monitors directly access video streams without the involvement of the PC [MILL99].

- IDS will have to take into consideration the adoption of security technologies such as Public Key Certificate Management, Smart Cards, and IPsec.

There are also a number of technologies on the horizon that will probably not be part of critical computing resources in the near future, but are worth considering in order to get a feel for how networking technology is emerging and for the direction of current research. Whether these technologies become successful or are replaced by other technologies remains to be seen, but they certainly will present an interesting challenge for network security.

- Wearable computing may be just around the corner. As people move through the world, the devices they wear or carry can provide their access portal to networked information, no matter where they roam. Similar visions exist for multi-device communications in the home and in the automobile. The interaction between such groups of embedded devices, which might be called "Smart Spaces," and the devices worn or carried into those spaces by people, could indeed produce a revolution in networking and computing as we conceive of it today. If "Smart Highways" ever become real, then the interaction between "Smart Highways", "Smart Cars", and "Smart Body LANs" could provide useful information, grounded in a relevant context at our fingertips on a continuous basis [MILL99].

- Pico-cellular wireless systems also offer some new and intriguing possibilities. Among the nearly 4.5 billion computer chips sold each year in the world, approximately 4 billion reside within embedded devices, such as microwave ovens, washing machines, and video cassette players. This trend is expected to accelerate, as computer chips find their way into more embedded devices, sensors, and actuators, and also into a growing number of portable and wearable devices expected to appear on the market over the next decade. To date, these chips, hidden within devices, have been largely inaccessible because no network capabilities have been included. With the advent of pico-cellular wireless technologies, such as Bluetooth and HomeRF, this situation is likely to change. Over the course of time, embedded devices will be able to communicate with each other, with their users, and with other computers and services connected to the Internet. These new capabilities should open astounding opportunities for improved automation in the home, office, and factory, as well as among mobile professionals [MILL99].

- Increasingly people work and live on the move. To support this mobile lifestyle, especially as work becomes more intensely information-based, companies are producing various portable and embedded information devices. Consider for example, PDAs will include embedded devices with high bandwidth, localized wireless communication capabilities that can also reach the globally wired Internet. A scenario of small, specialized devices roaming among pockets of wireless connectivity within a global wired networks is envisaged. Each wireless pocket becomes a "Smart Space", where available services and embedded devices can be discovered, accessed, interconnected with portable devices carried onto the island, and then the combination of imported and native devices can be exploited to support the information needs of the current network users [MILL99].

- Increasing computer density will allow mobile users to form ad hoc public networks. A "mobile ad hoc network" is an autonomous system of mobile routers (and associated hosts) connected by wireless links--the union of which form an arbitrary graph. The routers are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Such a network may operate in a standalone fashion, or may be connected to the larger Internet. [http://www.ietf.org/html.charters/manet-charter.html]

- Active networks are a novel approach to network architecture in which the switches of the network perform customized computations on the messages flowing through them. Active networks make use of intelligent packets that are no longer just data bits, but contain mobile code that allows for the active participation in routing, fault-tolerance, and quality of service decisions. This approach is motivated by both user applications, which perform user-driven computation at nodes within the network today, and the emergence of mobile code technologies that make dynamic network service innovation attainable. Active networks permit a significant improvement in the sophistication of the computation that is performed within the network. They will enable new applications, especially those based on application-specific multicast, information fusion, and other services that leverage network-based computation and storage. Furthermore, they will accelerate the pace of innovation by decoupling network services from the underlying hardware and allowing new services to be loaded into the infrastructure on demand [TENN97].

Appendix B - Architectural Issues

The first generation of intrusion detection systems followed a simple two component architecture: collection component and analyzer component. The collection component gleans information either from audit logs and internal interfaces at the host [SMAH88, LUNT88, SEBR88] or from monitoring packets on attached networks [HEBE90]. That information then feeds into a centralized analysis component, which employs one or more different detection techniques. The two logical components are either collocated at a single host or physically distributed. While this architecture is effective for small collections of monitored hosts, centralized analysis limits the ability to scale up to handle larger collections. Subsequent generations of IDSs address scalability mainly by introducing intermediate components between the collection and analysis components to form a hierarchy. This appendix provides a brief overview of the architectural and design issues associated with hierarchically structured IDSs when compared with network architectures, and how mobile agent technology applies to both.

Hierarchical Organization

A hierarchical architecture follows a tree structure with command and control components at the top, information aggregation units at the internal nodes, and operational units at the leaf nodes. The operational units can be network-based IDSs, host-based IDSs, virus checkers, and attack response systems. Figure 1 illustrates this architecture, which is followed by nearly all present day commercial IDSs. The circles denote individual nodes within a network and the arrows indicate information flow between different types of nodes.

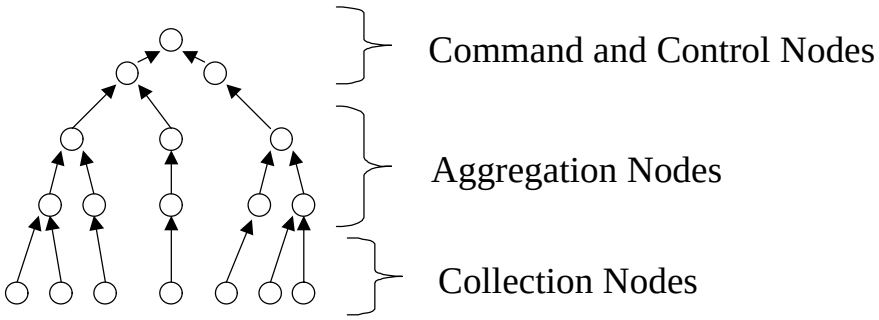

Figure 1: Hierarchical Intrusion Detection Architecture

Information gathering occurs at the leaf nodes. That information is passed to an internal node that aggregates information from multiple leaf nodes. Further aggregation, abstraction, and data reduction occurs at higher internal nodes until the root is reached. The root is a command and control system that evaluates attack situations and often issues responses. The root usually reports to human user consoles that can manually issue responses and evaluate the network.

Hierarchical structures result in efficient communications, whereby refined information filters upward in the hierarchy and control downward. The architecture is excellent for creating scalable distributed IDSs with central points of administration. These structures are, however, somewhat rigid in nature because of the precise function and lines of communication that tend to become associated with their components.

Network Organization

In contrast to a hierarchical architecture, a network architecture permits information flow from any node to any other node. Therefore, networked structures tend to suffer from inefficiency in communications, because of the unconstrained communication flow (i.e., everyone trying to communicate with everyone else). They do, however, compensate for communications inefficiency with flexibility in function. At least one IDS, Cooperating Security Managers [WHIT96], uses this architecture by consolidating the collection, aggregation, and command and control functions into a single component residing on every monitored system. Any significant events occurring at one system that stem from a connection originating at another are reported back to the system manager of the originating system by the security manager at the system where the event occurred. In situations where the originating system of the connection is an intermediate node in a communication chain, the system manager is obliged to report onward to the next system manager in the chain.

Implicitly IDS components tend toward a hierarchy, however, the tendency is not strict, since communications can occur, in general, between any type of components and not strictly on a one-to-one or master/slave basis. For example a collection unit may directly communicate a critical event to command and control unit, rather than through an aggregation node. Moreover, peer relationships among command and control nodes are appropriate where different administrations manage portions of an enterprise network or distinct and separate networks [FRIN98].

As a way of incorporating the best characteristics of hierarchical and network architectures, a hybrid model can be used. A hybrid model follows a network architecture. It has no distinct root, yet retains an overall hierarchical structure, and allows the flexibility for components to communicate outside the strict hierarchy, where useful. Figure 2 gives an example of this architecture, which illustrates peer relationship among command and control components, direct communication between collection units and a command and control unit (e.g., an event trigger), and redundant communication between an aggregation unit and command and control unit (e.g., for fault tolerance).

With mobile agents, the collection nodes, aggregation nodes, and command and control nodes do not have to be continuously resident on a physical machine. That is, a mobile agent may function as an aggregation node and move to whatever physical location in the network is best for its purposes. In fact, a mobile agent architecture offers an additional refinement on this idea. - the agents may be different for different functions. For example, there may be a hierarchy of mobile agents dedicated to detecting and responding to viruses. The aggregation and command and control nodes required for

virus detection might be totally different mobile agents than those required for, say, insider auditing. Thus, many hierarchies of agents may exist, each looking for different attacks and each processing data differently.

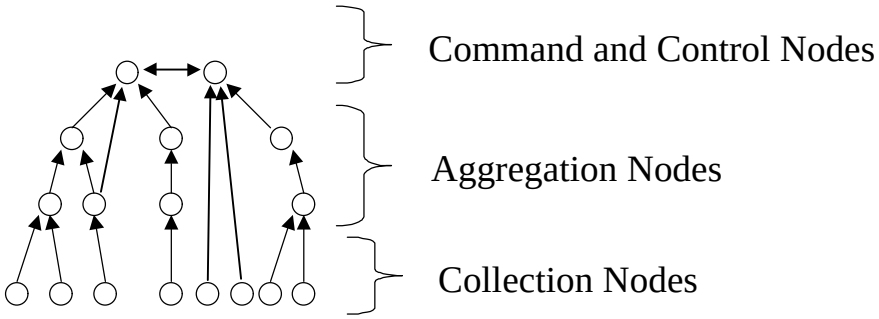

Figure 2: Hybrid Intrusion Detection Architecture

Framework for Intrusion Detection

The Common Intrusion Detection Framework [CIDF] provides a useful perspective for understanding and discussing the features and components found in any IDS. The CIDF defines the following generic types of components:

- *Event Generators* - audit data filters, burglar alarms, or other sensors used to obtain events from the computational environment.

- *Analyzers* - components such as an event filter, an attack signature detector, a statistical event profiler, or an event correlator, used to obtain intrusion detection information (IDI) from other components, analyze them, and return new IDI. IDI includes events that occurred in the system, analysis of those events, prescriptions to be carried out, or queries about events.

- *Databases* - components that perform no processing of or changes to the information they retain, but simply represent persistent storage of IDI.

- *Response Units* - components that carry out prescriptions from other components. Prescriptions are requests, such as killing processes or resetting connections, instructing response units to act on behalf of other components.

- *Matchmaker Units* – components that provides the configuration and directory services which link other components together. A matchmaker allows components to locate collaborating components either by name or by service.

Components may be organized in either hierarchical, network, or hybrid structures. Components may also support either a push or pull style of interface. The former refers to the production of IDI spontaneously by the component, while the latter refers to IDI produced in response to a request or query. Some components may also be further

decomposed. For example, an analyzer may be realized by two distinct components: an inference agent, which deduces intrusions, and a decision or planning agent, which formulates responses. Similarly, one or more components may be combined together and collocated at the same node. However, the initial types of components defined are adequate to capture the essential features of an IDS.

Any of the CIDF components can be represented as a mobile agent. However, some components are better suited if they remain stationary and may be designated as static agents. For example, an agent wrapper applied to a database management system could implement the CIDF database component. This pragmatic approach allows both the software agent paradigm to be maintained and the design to apply mobility where appropriate. It is also unlikely that, in practice, full mobility of all components would ever be effective. The following paragraphs describe some of the benefits and drawbacks of using mobile agents for common IDS components.

- *Network Traffic Sensors*: Keeping up with increasing network traffic seems to be becoming the job of the kernel or special purpose hardware. A mobile network traffic sensor, if it ever could keep up with network traffic, would lose information when it packs up and goes to another machine. The mobile agent wouldn't be well suited to directly monitor network traffic, but only data provided by the network sensor to the data analyzers.

- *Host Based Sensors*: Mobile agents could go to hosts (e.g., workstations, firewalls, routers, etc.) to collect information that is not available on the wire. Where they go can be in response to the analysis of audit data filters or the triggering of network trip wires.

- *Analyzers*: Analyzer agents process audit logs, correlate data from distributed network traffic monitors and process data gathered by network sensors. As new attacks are recognized or new patterns of suspicious activity are recognized, the analyzers must be upgraded while others removed as they are no longer useful. Analyzer agents can perform specialized data searches that aren't supported by the host platform on which the data resides.

- *Responders*: Mobile agents are well-suited to serve in the capacity of responder agents. These agents can travel through the network to reconfigure the network especially useful for thin clients that do not possess the functionality required to make decisions and carry out the administrative changes.

- *Coordinators:* Agents with the ability to decompose and solve problems in a collaborative fashion have been developed successfully by the intelligent agent community for a number of domains. Agents observe, reason, interact with other agents, and execute actions concurrently with other agents. Interactions may convey facts or beliefs via an agent communication language and may depend on ontologies to reach a common understanding.

The various scenarios raise the question, how these various characteristics can be integrated under a single MAIDS framework or design? This question will need to be addressed, either implicitly or explicitly, because of the value such a perspective has in constructing any MAIDS implementation.

Design Tradeoffs

In principle, mobile agent technology allows a node to taken on any of the various functions of an IDS. While mobility is an important focus of our project, when developing a design or performing a reification of the model for a specific network configuration, some components may be bound to a specific device as a static agent or restricted to a set of devices, when and where appropriate. Such engineering tradeoffs are common in the design of any IDS, and motivated by a number of factors including:

- Trust relationships among platforms,
- Platform performance considerations, and
- Physical location of a platform

For example, it might be reasonable to relegate a few platforms with dedicated communications among them, exclusively for use by the IDS. This action would ensure that critical communications and processing of IDSs is not affected by the overall workload on the system, and similarly that the IDS processing has a negligible impact on the system workload. Moreover, these platforms could implement a high standard of security and be hardened against penetrations to ensure a stable baseline for critical processes. In this example, confining critical or high consumption processes to a set of dedicated platforms is a tradeoff of mobility for performance and assurance.

An important aspect is to ensure that the introduction of mobility maintains a comparable or improved level of assurance and performance in the design. For example, [FRIN98] points out that within classical hierarchical IDSs, trust relationships are strong in the downward directions (i.e., subordinates trust superiors), but weaker in the reverse (i.e., superiors don’t trust subordinates). This is due to the fact that critical components, high up in the hierarchy reside on hardened systems, which are resistant to attacks. The notion of making every IDS node a mobile agent platform to which any agent component could visit would obviously weaken the original trust relationships, unless all of the platforms were hardened to the highest level. However, as long as the agents move among hosts within the same security domain and each security domain places the agent under the same security policy, granting or denying the same privileges it had on its previous platform, security concerns do not adversely affect mobile agents' ability to dynamically adapt to the execution environment.