Mobile Agents in Intrusion Detection System: Review and Analysis

Kamaruzaman Maskat, Mohd Afizi Mohd Shukran, Mohammad Adib Khairuddin & Mohd Rizal Mohd Isa
Faculty of Science and Technology Defense, Department of Computer Science
National University Defense University of Malaysia
Sungai Besi Camp 57000 Kuala Lumpur, Malaysia
{kamaruzaman, afizi}@upnm.edu.my

Received: August 26, 2011 Accepted: September 19, 2011 Published: December 1, 2011
doi:10.5539/mas.v5n6p218 URL: http://dx.doi.org/10.5539/mas.v5n6p218

Abstract

Intrusion Detection System (IDS) is used to detect intrusion and then alert the system administrator about the intrusion. This is what traditional IDS is all about. It is then up to the system administrator to deal with the intrusion. Human intervention is still needed when it comes to dealing with intrusion. This is because traditional IDS could only detect the intrusion but could not, on its own respond towards the intrusion. IDS is only able to alert the system administrator when it detects an intrusion. How and when the intrusion is dealt with is up to the system administrator. Human intervention when dealing with intrusion is not a problem if the person assigned to that task is always reliable. Therefore, this paper analyzes the evolution of IDS and how mobile agents such as SNORT could increase the integrity of traditional systems without human intervention.

Keywords: Intrusion Detection System, Mobile agents, Network security

1. Introduction

This paper starts off with some definitions of Intrusion Detection Systems (IDS), brief history of IDS, categories of IDS, IDS architecture and what is Snort. Then this paper will continue with the problem of traditional IDS that is the single point of failure and followed by the significant of mobile agents, agent based IDS, intelligent mobile agent, intelligent agent based IDS.

2. Intrusion Detection Systems (IDS)

2.1 Definition

From the name itself, people could interpret that an IDS is a system used to detect intrusion. But still what an IDS actually does and what it really is, remains unclear. To better understand what IDS is, is to look at the definition of IDS. Peter Loshin (2001) defined IDS as:

"Intrusion detection is the art and science of sensing when a system or network is being used inappropriately or without authorization. An intrusion-detection-system (IDS) monitors system and network resources and activities and, using information gathered from these sources, notify the authorities when it identifies a possible intrusion."

Wikipedia (2002) on the other hand defined IDS as the following:

"An intrusion detection system (IDS) inspects all inbound and outbound network activity and identifies suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a system. "From the two definitions stated above, it clearly shows that IDS function is different from other types of security devices such as firewall, intrusion prevention system (IPS), and so forth. The next question which arises is whether IDS comes in the form of hardware or software. This question is answered by Rebecca Base and Peter Mell (2000) as they defined IDS as:

"Intrusion detection systems (IDSs) are software or hardware systems that automate the process of monitoring the events occurring in a computer system or network, analyzing them for signs of security problems."

Therefore an IDS could be defined as a hardware or a software that monitors all inbound and outbound network activities, resources and systems automatically and analyzing them for any inappropriate or unauthorized use and also for any suspicious patterns or signs of security problems that might indicate possible intrusion.

Section 2.2 is about IDS brief histories, that is, who are the persons responsible for introducing HIDS, anomaly detection and NIDS.

2.2 Intrusion Detection System (IDS) Brief History

According to Paul Innella in her article, The Evolution of Intrusion Detection System (http://www.securityfocus.com/infocus/1514) dated 12 November 2001, the first person to come up with the concept of IDS is James Anderson. In his paper titled Computer Security Threat Monitoring and Surveillance which he wrote at the beginning of 1980, Anderson stressed that his main purpose is to improve the computer security auditing and surveillance capability of a customer's system. James Anderson suggested that with the use of audit trail, any unauthorized access to files could be detected because audit trail is present in most machines. Thus, the audit trail will be used to monitor users. The audit records will then be sorted out according to the job identity, date, time, and so on. A session of record builder which will look into what type of audit data is thus collected and if possible the type of system used. Lastly, the surveillance program will be developed. From what James Anderson has proposed, it seems like the concept he introduced back then is similar to the current Host Based Intrusion Detection System (HIDS), where each host has its own IDS that will monitor user's activities. Therefore it is true as what Paul Innella said about James Anderson work that is; he is the one that started the HIDS and IDS in general.

Intrusion Detection Expert System (IDES) was developed by SRI International and Dr Dorothy E. Denning between the year 1983 and 1984. The concept of IDES is to monitor system's audit record and look for any abnormal patterns of system usage in order to detect any computer misuse or abuse. Examples of abnormal patterns stated by Dr Dorothy E. Denning in her paper, An Intrusion Detection Model, are shown below:

- *Attempted break-in:* Someone attempting to break into a system might generate an abnormally high rate of password failures with respect to a single account or the system as a whole.
- *Masquerading or successful break-in:* Someone logging into a system through an unauthorized account and password might have a different login time, location, or connection type from that of the account's legitimate user. In addition, the penetrator's behavior may differ considerably from that of the legitimate user in particular. The penetrator might spend most of his time browsing through directories and executing system status commands, whereas the legitimate user might concentrate on editing or compiling and linking programs. Many break-ins have been discovered by security officers or other users on the system who have noticed the alleged user behaving strangely.
- *Penetration by legitimate user:* A user attempting to penetrate the security mechanisms in the operating system might execute different programs or trigger more protection violations from attempts to access unauthorized files or programs. If his attempt is successful, he will have access to commands and files not normally permitted to him.
- *Leakage by legitimate user:* A user trying to leak sensitive documents might log into the system at unusual times or route data to remote printers not normally used.
- *Inference by legitimate user:* A user attempting to obtain unauthorized data from a database through aggregation and inference might retrieve more records than usual.
- *Trojan horse:* The behavior of a Trojan horse planted in or substituted for a program may differ from the legitimate program in terms of its CPU time or 1/0 activity.
- *Virus:* A virus planted in a system might cause an increase in the frequency of execuTable files rewritten, storage used by execuTable files, or a particular program being executed as the virus spreads.
- *Denial-of-Service:* An intruder who is able to monopolize a resource (e.g., network) might have abnormally high activity with respect to the resource, while activity for all other users is abnormally low.

Comparing normal system usage with abnormal system usage that Dr Dorothy E. Denning introduced to IDES is similar to the Anomaly techniques used by today's IDS where the normal behavior of the network is recorded and then compared with any abnormalities that occurred during the usage of the network. The first Network Intrusion Detection System (NIDS) was introduced by Todd Heberlein in 1990. The IDS is named Network Security Monitoring (NSM). NSM concepts are collecting, analyzing and rising indications and warnings in order to detect and respond to any intrusions. To summarize what has been said earlier, the first HIDS was introduced in 1980 by James Anderson, and then between 1983 and 1984 an anomaly detection technique was introduced by Dr Dorothy E. Denning. In 1990 IDS has taken another step forward when NIDS was introduced by Todd Heberlein. Section 2.3 will explain the categories of IDS which has been divided into three categories which are; information, analysis and response.

2.3 Categories

Now days there are different types of IDS, so the easiest way to understand them is to categorize them into three groups; first is information, second is analysis and third is response.

a. Information

Under this section, IDS is categorized according to the source that the IDS uses to gather information from. Basically there are two sources that IDS will use for collecting information; first is the network and second is the host. Przemyslaw Kazienko & Piotr Dorosz (2004) explained that, NIDS will use the network to congregate, and reassemble all network packets from the network. This is done by conFigureuring the network interface card to function in the promiscuous mode. NIDS could even monitor the network traffic of a network segment that has multiple hosts connected to it. Examples of NIDS are such as Cisco Secure IDS or NetRanger, Hogwash, Dragon, E-Trust IDS, and many more. HIDS is where the IDS reside in a host, so every host has its own IDS. HIDS will only examine traffic within a particular host by gathering information from the host's system calls, application logs, database, operating system audit trails, and so forth. HIDS could even function in an environment where the network traffic is encrypted (Wikipedia, 2006). Both NIDS and HIDS could be used together in order to protect and strengthen security. But for the purpose of this study NIDS is chosen because this study concentrates on network security and monitoring.

b. Analysis

Once IDS has harvest information from a network or a host, then the information will be analyzed. There are two methods used for analysis; first is misuse and second is anomaly. Misuse of analysis is similar to the way antivirus works that is by having a huge database of known attacks. The IDS will then compare the data collected with that of the database. If there is a match then the IDS will assume an attack has occurred. Misuse is sometimes called signature-based IDS because the pattern contained in the database is called signatures (Wikipedia, 2006). In an anomaly method it is assumed that attacks or threats are different from the normal usage or behavior of a host, network or network connection. Hence any abnormal behavior (anomalies) or usage of a host, network or network connection will cause the IDS to identify it as an attack (Rebecca Base and Peter Mell, 2000). Misuse method doesn't give lots of false alarm compared to Anomaly method because if the data collected doesn't match the signature of known attacks then no alarm will be produced. Unlike Anomaly method where if a user is doing something which is not of his or her normal activities such as chatting, a false alarm will be generated because the harmless abnormal activity is considered a threat by the IDS. Due to the reason of false alarm stated above, misuse method will be used for the purpose of this study.

c. Response

After IDS has gathered and analyzed data or information and detect an intrusion the IDS will respond to the intrusion in two ways; first is passive and second is reactive. In passive reaction, once an intrusion is detected the IDS will log the information about the intrusion and then originate an alert. Passive IDS rely on humans to take action when an intrusion occurred. Types of alert a passive IDS will produce are; alarms, notification, simple network management protocol (SNMP) traps and plug-ins (Wikipedia, 2006). Reactive IDS will response to intrusion via automated action set by the user. Examples of automated actions are; logging of users, block services, reprogramming firewalls to block certain IP addresses, and so on (Wikipedia, 2006). It's up to the user to choose which type of IDS reaction he or she needs. A passive reaction will need system admin attention most of the time in order to react towards intrusion, whereas reactive reaction will respond to intrusion autonomously with less human intervention. Passive reaction is preferred for this study because the IDS is needed to fully concentrate on detecting intrusion.

2.4 IDS Architecture

Basically IDS has two types of architecture; first is centralized and second is distributed. In a centralized architecture, all activities such as monitoring, gathering, analyzing, detecting, reporting, and so forth are all done by a central location. This central location will do all of the mentioned tasks for the whole network all by itself. Distributed architecture is when multiple IDS are used to monitor large network where monitoring and detection is being controlled by a local control node which will report to one or multiple central locations in hierarchical manner. All multiple IDS communicate with each other. Centralized architecture is vulnerable to attack because if the central location is down, then the entire network is open to attackers. Distributed architecture is more reliable because multiple IDS are used so that if one IDS is down the are backups, due to this reason distributed environment is used for this study.

2.5 Snort the De Facto Standard for IDS

a. What is Snort?

Snort is an NIDS; it is considered as "lightweight" intrusion detection compared to any other commercially available systems. It has become the present standard for intrusion detection system as stated by Snort.org:

"In 1998, Martin Roesch wrote an open source technology called Snort, which he termed a "lightweight" intrusion detection technology in comparison to commercially available systems. Today that moniker doesn't even begin to describe the capabilities that Snort brings to the Table as the most widely deployed intrusion prevention technology worldwide. Over the years Snort has evolved into a mature, feature rich technology that has become the de facto standard in intrusion detection and prevention. Recent advances in both the rules language and detection capabilities offer the most flexible and accurate threat detection available, making Snort the "heavyweight" champion of intrusion prevention."

b. Components of Snort

Snort consists of various components that work together to detect intrusion and to produce output in certain format. As stated by Rafeeq Ur Rehman (2003), Snort could be divided into five major components (Table 2.1) whereas Figure 2.1 illustrates how snort's components interact with each other:

c. Why Snort?

Snort is the NIDS chosen for this study because Snort has proved itself, time and time again, as the best NIDS there is in the market. There are a lot of issues regarding Snort's performance, for example the issue of evasion attacks. Evasion attacks are such as fragmentation reassembly timeout attack, TTL based attacks and overlapping fragments. All these attacks are not easily overcome by IDS but with Snort these attacks are detecTable as stated by Sumit Siddharth (2005):

"we've looked at a few IDS evasion attacks and the different methodologies involved with various attack. We have shown that it is important to know how different operating systems perform fragmentation reassembly. We finally conclude that these evasion attacks can be prevented by setting the frag3 preprocessor in Snort properly."

Snort is so flexible that it could even be used as intrusion prevention as claimed by Wikipedia (2006):

"It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more. The system can be used for intrusion prevention purposes too. Snort can also be combined with other open source projects such as SnortSnarf, sguil, and the "Basic Analysis and Security Engine" (BASE) to provide a visual representation of intrusion data."

Illija Basicevic, Miroslav Popovic and Vladimir Kovacevic (2005) proposed the use of Distributed Network-Based IDS Systems in detecting evasion attacks. Their suggestion is to use both the concept of host based and network based IDS in distributed environment, where they implement the NIDS at each host (client/host based) which will communicate with the main NIDS implemented in the network environment (network based). The NIDS that they used is Snort because it supports the use of algorithms such as Wu-Manber, Aho-Corasick and Boyer-Moore. Mohamad Eid (2005) used Snort in his research because it is a full-fledged open-source network based IDS (NIDS) with many capabilities such as sniffing, packet logging and intrusion detection. The concept here is, Snort as a distributed IDS, will control remote sniffers via mobile agents.

2.6 Weaknesses in Traditional IDS

In a centralized architecture there exists a single point of failure, that is, if the IDS fail due to attacks, breakdowns or other reasons, the whole network is vulnerable to attacks because attackers could sneak into the network undetected as mentioned by Mohamad Eid (2005):

"A single machine monitors data flow at a strategic point in the network and collects and analyzes data from the log files. Once an attacker destabilizes this host, he or she is able to gain considerable access to the whole network."

Another problem with centralized architecture is stated by Anita K. Jones and Robert S. Sielken (1999); centralized approach works well for smaller network situations but is inadequate for larger networks due to the sheer volume of audit data that must be analyzed by the central analysis component. James Fell (2002) also came across the same issue by stating that a problem with having a fixed centralized host for IDS analysis is that the bigger the network is, the more power is demanded of this host, making it impractical for large networks. But Mark J. Crosbie and Benjamin A. Kuperman (2001) argued that a centralized architecture offloads the processing and analysis components to a centralized location to reduce the load on the monitored nodes, which is true, but only for small networks, this is because a huge network will have heavy traffic and there will be a lot of packets to be monitored and analyzed and this will in turn cause a lot of delayed or queued data to be processed by the centralized location. Columbia University DNAD Team (2005) assets which is a centralized system, cannot aggregate and correlate information quickly enough to provide system administrators a reasonable chance at reaction. Therefore other architecture has to be considered to protect large network from these weaknesses. Other architecture which is able to overcome this problem is Distributed IDS (DIDS). Nathan Einwechter (2002) defined DIDS as multiple Intrusion Detection Systems (IDS) over a large network, where all of which communicate with each other, or with a central server that facilitates advanced network monitoring, incident analysis, and instant attack data. Early DIDS are such as ASAX (Mouinji, 1995), DIDS (Snapp, 1999) and NSTAT (Kemmerer, 1997), NetSTAT (Vigna, 1999), and many more. These systems collect information using distributed method but the analysis is still being done centrally. Hence, the single point of failure still exists even though this design is much better than centralized IDS architecture.

A hierarchy approach where lower distributed IDS will report to higher IDS is the next evolution in DIDS where this architecture objective is to demolish the single point of failure problem. Examples of DIDS using such approach are EMERALD (Porras, 1997), GriDS (Staniford, 1996) and AAFID (Spafford, 2000). Unfortunately the problem of single point of failure has not been overcome entirely and this is because of the existence of the highest single IDS in the hierarchy architecture. Other weaknesses arise from the hierarchy approach as mentioned by Ajith Abraham and Johnson Thomas (2005); the main problem with such an approach is that if two or more IDS that are far apart in the hierarchy detect a common intruder, the two detection cannot be correlated until the messages from the different IDS reach a common high-level IDS. This will require the messages to traverse multiple IDS resulting in communication overheads. The task now is to solve the hierarchy and the single of failure problem. There are several suggestions in using mobile agents to solve the hierarchy problem.

2.7 Mobile Agent

This section will explain the definition of mobile agent, the characteristics of software agent and the advantages of mobile agent.

2.7.1 Definition

Wikipedia (2006) defined software agent as a software abstraction, an idea, or a concept, similar to Object Oriented Programming (OOP) terms such as methods, function and objects. The concept of an agent provides a convenient and powerful way to describe a complex software entity that is capable of acting with a certain degree of autonomy in order to accomplish tasks on behalf of its user. But unlike objects, which are defined in terms of methods and attributes, an agent is defined in terms of its behavior.

So a software agent is something like a program that on behalf of a user could do or complete tasks for the user autonomously and it also has certain behavior or criteria. Section 2.7.2 will explain about software agent behavior.

2.7.2 Mobile Agent Behavior

National University of Singapore (2001) listed the behavior of an agent as shown below:

2.7.3 Mobile Agent

a. Definition

Mobile agent is defined by Wikipedia (2006) as a composition of computer software and data which is able to migrate (move) from one computer to another autonomously and continue its execution on the destination computer. Criteria of mobile agent is given by Todd Sundsted (1998) as autonomous, adaptive/learning, mobile, persistent, goal oriented, communicative/collaborative, flexible and active/proactive. IBM Tokyo Research Laboratory (TRL) (2002) defined mobile agents as programs that can be dispatched from one computer and transported to a remote computer for execution. Arriving at the remote computer, they present their credentials and obtain access to local services and data. The remote computer may also serve as a broker by bringing together agents with similar interests and compatible goals, thus providing a meeting place at which agents can interact.

b. Why Mobile Agent

Mobile agent offers several advantages when it comes to network management applications and solve several problems as listed by AV Scott (2000):

- Network Bandwidth
- Protocol Communications

- Intermittent Connections
- Load Balancing
- Real-Time Notification and
- Heterogeneous Networks

Danny Lang and Mitsuru Oshima (1998) and Jonathan Smith (1988) have also proposed a number of advantages of mobile agents as shown below:

- Overcoming network latency
- Reducing network load
- Executing asynchronously and autonomously
- Adapting dynamically
- Operating in heterogeneous environments and
- Robust and fault-tolerant behavior

Mobile agent has been proposed to overcome the hierarchy problem. This is because hierarchy architecture consists of command and controls components as the highest position in the hierarchy, second highest is the aggregation nodes and the lowest position is the multiple leaf nodes. The leaf nodes will gather or collect information and then pass it over to the aggregation nodes where further analysis and responds will be done by the command and controls components. With the use of mobile agents the problem mentioned by Ajith Abraham and Johnson Thomas (2005) earlier could be solved as stated by Jai Sundar Balasubarmaniyan et al (1998); by organizing the agents in a hierarchical structure with multiple layers of agents reducing data and reporting it to the upper layers, the system can be made scalable. This idea is explained in detail by Wayne Jansen et al (1999); 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. Mobile agent could move around the network and execute whatever tasks assigned to it. Due to its mobility, mobile agents are not an easy target for attackers. It will be very hard yet not impossible for attackers to kill mobile agents. The ability of mobile agents to stay away from areas where the attacker is attacking makes it harder for attackers to detect and kill agents. Mobile agents could communicate with other agents. They could even continue to function or operate, even after the host that launched them has been removed from the network. This is very beneficial in security because if IDS has been attacked by an attacker, mobile agents are still alive and keep on roaming the network whilst completing its task because mobile agents are also goal-oriented.

2.7.4 Aglets

a. What is an aglet?

The aglet represents the next leap forward in the evolution of execuTable content on the Internet, introducing program code that can be transported along with state information. Aglets are Java objects that can move from one host on the Internet to another. That is, an aglet that executes on one host can suddenly halt execution, dispatch itself to a remote host, and resume execution there. When the aglet moves, it takes along its program code as well as its data (IBM Tokyo Research Laboratory, 2002).

b. Aglet Object Model

The aglet object model describes all the abstractions and behaviors necessary to implement Java mobile agents. Figure 2.2 shows the major interfaces and classes defined in J-AAPI and the relationships between these. Table 2.2 shows the definition of terms used in Figure below (AV Scott, 2000).

c. Aglet Life Cycle

Once an aglet has been created, it could be clone, dispatch, retract and dispose. Figure 2.3 below shows aglet's life cycle (Mitsuru Oshima et al, 1998).

d. Aglet Architecture

The Aglet architecture consists of two APIs and two implementation layers as shown in Figure 2.4 below:

The Aglets runtime layer is the implementation of Aglet API, which provides the fundamental functionality such as creation, management or transfer of aglets. The transport layer is responsible for transporting an agent to the destination in the form of a byte stream that contains class definitions as well as the state of the agent. The current

Aglets implementation uses the Agent Transfer Protocol (ATP), which is an application-level protocol for transmission of mobile agents. ATP is modeled on the HTTP protocol, and can be used to transfer the content of an agent in an agent-system-independent manner. To enable communication between agents, ATP also supports message passing (Marcus Naylor, 2000).

e. Why Aglet?

Aglet is chosen for this study because it is simple and it follows an applet like development paradigm, it is also flexible because the platform could be extended in order to implement new functionalities, and the most important thing is, it is freely available (sourceforge.net, 2004). Aglet's characteristics (Jagadha Sivan, 2000) as shown in Table 2.3 below are suiTable for this study due to the distributed environment proposed to overcome weaknesses that exist in traditional IDS:

Aglet also has security features where it uses an organizational approach whereby all agent systems in a certain domain are deemed trustworthy and evaluates the authenticity of the agent depending on the domain in which it has been roaming around. A user first authenticates himself or herself to the system, and the system then issues the credentials of the user's agent. The agent system then evaluates the authenticity of the credentials, to determine whether or not they were issued within the same domain. It may downgrade the authenticity or simply deny access, depending on conditions such as where the agent has traveled and so forth. Host authentication is used to identify the domain to which the communicating host belongs (Mitsuru Oshima et al., 1998).

2.8 Agent Based IDS

Since agent opened up a new era in intrusion detection, a lot of effort has been made to integrate agent with IDS. D'Agents proposed by Jay Aslam et al. (2002), consists of four components; the basic agent infrastructure, a module for the distributed capturing and accessing of security logs, a module for correlating data from the security logs and a module for formulating attack hypotheses based on the output of the correlation module. The objective of D'Agents is to be the automated tools for computer experts and system administrators to be able to; identify the characteristics of an attack given data from network sensors, develop a hypothesis about the nature and origin of the attack, share that hypothesis with security managers from other sites, test that hypothesis at those other sites and coordinate the results of testing and archive the data necessary for use as evidence in later law-enforcement actions. Because of the ability of traveling from one point to another, the mobile agent is used to travel to the location of the system logs, filter out the time-relevant or location-relevant information from these logs and correlate all this information without necessitating costly file transfers across the net. (response-stop intrusion in the early stage by isolating the compromised host).

To overcome the single point of failure vulnerability and to determine where networks attacks are coming from are the objectives of D.Frincke and E.Wilhite (2001) in integrating mobile agents with IDS. The mobile agents are divided into three categories; tool agents, investigation agents and defensive agents. These agents are sent to network nodes to monitor or seek out a specific network or host-based event in order to gather a wider array of information that will be used in their response activities- in future to directly manage host/network's defensive posture.

Peter Mell and Mark McLarnon (1999) found that a distributed intrusion detection systems are vulnerable to attacks because each components resides at a static location and connected together into a hierarchical structure. An attacker can disable such a system by taking out a node high in the hierarchy. To solve this problem, Peter Mell and Mark McLarnon (1999) proposed the use of mobile agents as the internal nodes in the system hierarchy due to the fact that mobile agents could randomly move around the network making it hard for attackers to locate their position. And even though if the mobile agent platform has been taken out by attackers, mobile agents will be able to estimate the location of the attacker and automatically avoid those networks. Furthermore killed agents are resurrected by a group of backups that retain all or partial state information-no autonomous respond towards intrusion, still need sys admin.

Traditional intrusion detection systems have a central coordinator with a static hierarchical architecture, which indicates the existence of the single point of failure and hierarchy vulnerability. Geetha Ramachandran and Delbert Hart (2004) proposed a solution of a peer-to-peer intrusion detection system with no central coordinator. Mobile agents are used to visit and check up on its neighbors and report back. If there's inconsistent or anomalous behavior the mobile agent will initiate a voting process to take action against the compromised site. Christopher Kruegel et al (2002) addressed the problem of system administrator having problem in making sure that his or her large network environment has no security holes. In conjunction with this problem Christopher Kruegel et al (2002) proposed the use of mobile agents to correlate event data that will help to detect security policy violations and network intrusions in a heterogeneous, networked environment. The use of mobile agent is also to increase the fault tolerance and scalability of the proposed system – no respond towards intrusion. When attackers attack an organization's network, system administrator will need all the help he or she can get in dealing and responding to the attack. The basic thing to do when attack occurred is to stop it as mentioned by Joseph Barrus and Neil C. Rowe (1997):

"When an intruder attacks a system, the ideal response would be to stop his activity before he can do any damage or access sensitive information."

If certain action such as stopping an attacker activity could be done automatically, this would certainly help to prevent any further damage. In order to do this Joseph Barrus and Neil C. Rowe (1997) suggest the use of mobile agents where a collection of autonomous agents running on the various hosts within a heterogeneous network, provides the foundation for a complete solution to the complexities of real-time detection and response in such an environment (Barrus, 1997). These agents as explained by Joseph Barrus and Neil C. Rowe (1997) will monitor intrusive activity on their individual hosts. Each agent can be conFigured to the operating environment in which it runs. In addition, other third party tools-- LAN monitors, expert systems, intruder recognition and accountability, and intruder tracing tools-- can be "plugged in" to augment the basic host-monitoring infrastructure. The agents communicate with each other relaying alert messages utilizing a hierarchical scheme of escalating levels of alertness. Agent Based IDS, as mentioned above, is the proposed solution to identify characteristics of an attack, overcoming single point of failure weaknesses in traditional IDS, overcoming weaknesses of the distributed IDS hierarchical structure, overcoming security holes in large network and also to stop attacker's activity autonomously.

2.9 Similar Project

Zahaoyu Liu and Roopesh Uppala (2006) from University of North Carolina proposed a similar project entitled A Dynamic Countermeasure Method for Large-Scale Network Attacks. This project uses Snort as the IDS and two types of agents; Snortsam and Gnipper vaccine. According to Zahaoyu Liu and Roopesh Uppala (2006), Snortsam is an intelligent agent that integrates with Snort to perform a block operation on a remote firewall. This allows Snort to block intruding connections by generation of dynamic ipTables firewall rules. Snortsam will request a block on firewall host where it resides. Gnipper vaccine is a dynamic agent that resides on a host and capable of dropping any malicious packets. It will propagate one hop at a time towards the source of the attackers thus disabling the ability of the attacker to ping the intended victim (Zahaoyu Liu and Roopesh Uppala, 2006). The concept of how this system works as explained by Zahaoyu Liu and Roopesh Uppala (2006) is, when a malicious packet, for example ICMP echo requests packet from a malicious ping request reaches a trusted network, Snort will detect it and the packet will be dropped by Snortsam and Gnipper vaccine will move from the trusted network towards the source of the malicious ping request one hop. All this occurred when the first packet of the ICMP echo requests reaches the trusted network. This process will repeat itself for each ICMP echo requests that reaches the trusted network, causing the Gnipper vaccine to move towards the source of the malicious ping request one hop at a time until it reaches the source and disabling the source ability to ping the trusted network. Attackers usually will use spoofed ip addresses, and if the attacker uses different spoofed ip addresses to ping the trusted network, then the trusted nodes will be full of Gnipper vaccine. This is because; the malicious ping packet will be detected by Snort which resides in the trusted network.

Moreover, Snortsam will then drop the packets and propagate Gnipper vaccine to the source of the malicious packets. So every time a new malicious ping packet with different ip addresses arrives at the trusted network, Gnipper vaccine will again and again repeat the same process of hopping from the trusted network towards the attacker. Even though agents are known of being lightweight, but if there are too many of them on the network and the numbers are increasing, the network might suffer congestion or decrease in bandwidth which could lead to denial of service. Moreover even though the process of Gnipper vaccine is reversible, it still needs the system administrator to reverse it. If the scenario mentioned above occurred, it will be troublesome for the system administrator to reverse all the processes.

3. Conclusion

Intrusion Detection System (IDS) in its traditional form is doing intrusion detection all by itself. Its function is to alert system administrator once an intrusion is detected. The system administrator then will take appropriate action to deal with the intrusion. For example, Agent Based Snort (ABS) can be used to discover whether the integration of IDS and Mobile Agent would lessen the human intervention when an intrusion is detected having the Mobile Agent autonomously respond towards the intrusion. Moreover, using mobile agents in IDS would increase the integrity of a database in which it would keep records of what type of intrusion the mobile agent responds to and what command it executes to instruct the firewall to block the intrusion. This information is important since the changes made to the firewall will disappear when the host restarts. It is also important for system administrator to know, analyze and monitor what the mobile agent has done.

4. References

Abraham, A., & Thomas, J. (2005). *Distributed Intrusion Detection Systems: A Computational Intelligence Approach.* Chung-Ang University.

Abraham, A., Grosan, C., & Chen, Y. H. (2005). *Cyber Security and the Evolution of Intrusion Detection Systems*. Chung-Ang University.

Albag, H. (2001). *Network & Agent Based Intrusion Detection Systems*. Istanbul Technical University.

Amal, E. F. S., & Alexandru, S. (2000). *An Unified Framework for Programming Autonomous, Intelligent and Mobile Agents*. University of Paris 6.

Aslam, J., Cremonini, M., Kotz, D., & Rus, D. (2001). Using Mobile Agents for Analyzing Intrusion in Computer Networks. *In Proceedings of the Workshop on Mobile Object Systems at ECOOP 2001*. July. Hanover, NH. http://www.cs.dartmouth.edu/~dfk/papers/aslam:position.pdf

Bace, R., & Mell, P. (2001). Intrusion Detection Systems. *Infidel, Inc., Scotts Valley, CA and National Institute of Standards and Technology.*

Balasubramaniam, J. S., Garcia-Fernandez, J. O., Isaoof, D., Spafford, E., & Zamboni, D. (1998). An Architecture for Intrusion Detection Using Autonomous Agents. *COAST Technical Report 98/05*. June 11. Purdue University.

Barrus, J. (1998). A Distributed Autonomous-Agent Network-Intrusion Detection and Response System. *Procedings of the 1998 Command and Control Research and Technology Symposium*. June-July. Monterey, CA. http://www.cs.nps.navy.mil/people/faculty/rowe/barruspap.html

Basicevic, I., Popovic, M., & Kovacevic, V. (2005). The Use Of Distributed Network-Based IDS Systems In Detection Of Evasion Attacks. *Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/ELearning on Telecommunications Workshop*. Montenegro: IEEE.

Bigus, P. J., & Bigus, J. (2001). *Constructing Intelligent Agents Using Java*. Canada: John Wiley & Sons, Inc.

Botha, M., Solms, R. V., Perry, K., Loubser, E., & Yamoyany, G. (2002). The Utilization of Artificial Intelligence in a Hybrid Intrusion Detection System. *Proceedings of SAICSIT 2002.* Port Elizabeth Technikon, 149–155.

Brennan, M. P. (2002*). Using Snort For a Distributed Intrusion Detection System Version 1*. SANS Institute.

Cardoso, R. C., & Freire, M. M. (2004). *Intelligent Assessment of Distributed Security in TCP/IP Networks*. University of Beira Interior.

Columbia University DNAD Team. (2005). *On The Feasibility of Distributed Intrusion Detection.* Columbia University.

Crosbie, M. J., & Kuperman, B. A. (2001). *A Building Block Approach to Intrusion Detection*. Hewlett-Packard Company,Purdue University.

Carver, Curtis, A., Jr., John, M.D. Hill, Member, John R., Surdu, IEEE, & Udo, W. (2000). Pooch, Senior Member, IEEE. A Methodology for Using Intelligent Agents to provide Automated Intrusion Response. *Proceedings of the 2000 IEEE Workshop on Information Assurance and Security*. June 6-7. West Point, NY: IEEE, 110–116.

Deeter, K., Singh, K., Wilson, S., Filipozzi, L., & Vuong, Son. (2005). *APHIDS: A Mobile Agent-Based Programmable Hybrid Intrusion Detection System*. University of British Columbia.

Duda, R., Hart, P. E., nillson, J. J., Reboh, R., Slocum, J., & Sutherland, G. (1997). *Development of A Computer-based Consultant for Mineral Exploration. In SRI Report*. Menlo Park, CA:Stanford Research Institute.

Eanes, M. (2003). *Wanted Dead or Alive: Snort Intrusion Detection System*. SANS Institute.

Frincke, D., & Wilhite, E. (2001). Distributed Network Defense. *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, June 5-6. West Point. NY: IEEE, 236–238.

Fukuta, N., Takayuki, I., & Toramatsu, S. (2002). *An Approach to Building Mobile Intelligent Agents Based on Anytime Migration*. Nagoya Institute of Technology.

Govindu, S. K. (2005). *Intrusion Forecasting System*. (http://www.securitydocs.com/library/3110)

Helmer, G. G., Johnny S. K., Wong, V. H., & Miller, L. (1999). *Intelligent Agents for Intrusion Detection*. Iowa State University.

Helmer, G. G., Johnny, S. K., Wong, V. H., Miller, L., & Wang, Y. X. (2003). Lightweight agents for intrusion detection. *The Journal of Systems and Software*, (67), 109–122. (www.elsevier.com/locate/jss)

Jansen, W., & Karygiannis, T. (1999). Mobile Agent Security. *NIST Special Publication 800-19*. National Institute of Standards and Technology.

Jansen, W., Mell, P., Karygiannis, T., & Marks, D. (1999). Applying Mobile Agents to Intrusion Detection and Response. *NIST Interim Report (IR) – 6416*. October. National Institute of Standards and Technology.

Johnny, S. K. W., & Mikler, A. R. (1997). Intelligent Mobile Agents In Large Distributed Autonomous Cooperative Systems. *The Journal of Systems and Software*. (47), 75–87. (www.elsevier.com/locate/jss)

Jones, A. K. & Sielken, R. S. (2000). *Computer System Intrusion Detection: A Survey*. University of Virginia.

Kark, K., McClean, C., Koetzle, L., Penn, J. & Bernhardt, S. (2007). *2007 Security Budgets Increase: The Transition to Information Risk Management Begins*.

Kruegel, C., Toth, T., & Kirda, E. (2002). *Sparta-A Mobile Agent based Intrusion Detection System*. Technical University of Vienna.

Krügel, C., & Toth, T. (2002). *Flexible, Mobile Agent Based Intrusion Detection for Dynamic Networks*. Technical University Vienna.

Kun, Y., Galis, A., Guo, X., & Liu, D. (2003). *Rule-Driven Mobile Intelligent Agents for Real-Time ConFigureuration of IP Networks.* University College London.

Lazareviæ, A., Srivastava, J., & Kumar, V. (2003). Data Mining For Intrusion Detection. *Tutorial on the Pacific-Asia Conference on Knowledge Discovery in Databases 2003*. University of Minnesota.

Liu, Z., & Uppala, R. (2006). *A Dynamic Countermeasure Method for Large-Scale Network Attacks.* University of Carolina.

Malakooti, B., Thomas, I. J., Bhasin. K., & Holtz. A. (2005). *A Framework for an Intelligent Internet Protocol for a Space-Based Internet*. Case Western Reserve University.

McDermott, J. (1982). R1: A Rule-Based CnFigurer of Computer Systems. *Artificial Intelligence*. 19(1),39-88. http://dx.doi.org/10.1016/0004-3702(82)90021-2

McHugh, J., Christie, A., & Allen, J. (2000). Defending Yourself: The Role of Intrusion Detection Systems. *IEEE SOFTWARE*. September/ October. IEEE, 42–51.

Mehdi, S., & Ghorbani, A. A. (2005). *Agent-oriented Design for Network Survivability*. University of New Brunswick.

Mell, P., & McLarnon, M. (1999). *Mobile Agent Attack Resistant Distributed Hierarchical Intrusion Detection Systems*. National Institute of Standards and Technology.

Mohamad, E. (2005). *A New Mobile Agent-Based Intrusion Detection System Using Distributed Sensors*. American University of Beirut.

Muller, N. J. (1997). Improving Network Operations With Intelligent Agents. *International Journal Of Network Management*. 7, 116–126. http://dx.doi.org/10.1002/(SICI)1099-1190(199705/06)7:3<116::AID-NEM239>3.0.CO;2-Q

Nardi, B. A., Miller, J. R., & Wright, D. J. (1988). Collaborative, Programmable Intelligent Agents. *Communications Of The ACM*. 41(3), 96–104. http://dx.doi.org/10.1145/272287.272331

Newman, C. R. (2006). Cybercrime, Identity Theft and Fraud: *Practicing Safe Internet – Network Security Threats and Vulnerabilities*. Georgia Southern University.

Pickering, K. J. (2001). *Evaluating The Viability Of Intrusion Detection System Benchmarking*. University of Virginia: Degree Bachelor Thesis.

Pillai, M. M., Eloff, J. H. P., & Venter, H. S. (2005). *An Approach to Implement a Network Intrusion Detection System using Genetic Algorithms*. University of Pretoria.

Qi, H., Xu, Y., & Kuruganti, T. P. (2004). *The Mobile Agent Framework for Collaborative Processing in Sensor Networks*. University of Tennessee.

Rafeeq, U. R. (2003). *Intrusion Detection Systems with Snort - Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID*. New Jersey: Prentice Hall PTR.

Ramachandran, G., & Hart, D. (2004). A P2P Intrusion Detection System based on Mobile Agents. *ACME '04*. April 2-3. Alabame, USA: ACM, 185–190.

Rudowsky, I. (2004). Intelligent Agents. *Proceedings of the Americas Conference on Information Systems.* August. New York: Brooklyn College.

Sielken, R. S. (1999). *Application Intrusion Detection*. University of Virginia: Master Thesis. http://www.cs.virginia.edu/~jones/IDS- research/Documents/MCS-9905-Sielken.doc

Sindhu, Sivatha, S, S., Ramasubramanian, P., & Kannan, A. (2005). *Intelligent Multi-agent Based Genetic Fuzzy Ensemble Network Intrusion Detection*. Anna University.

Sun, B. (2004). *Intrusion Detection In Mobile Ad Hoc Networks A Dissertation.* Texas A&M University: Ph. D Thesis.

Ulieru, M. (2005). Network and Information Security Workshop. *Adaptive Risk Management System (ARMS)*. Halifax.

Vigna, G., Valeur, F., & Kemmerer, R. A. (2003). Designing and Implementing a Family of Intrusion Detection Systems. *ESEC/FSE'03.* September 1–5. Helsinki, Finland: ACM, 88–97.

Vuong, S. T. & Fu, P. (2002). *A Security Architecture and Design forMobile Intelligent Agent Systems*. University of British Columbia.

Wasniowski, R. A. (2005). Multi-Sensor Agent-Based Intrusion Detection System. *Information Security Curriculum Development (InfoSecCD) Conference '05.* September 23-24. Kennesaw, GA: ACM, 100–103.

Webopedia. (2002). Intrusion Detection Syste (http://www.webopedia.com/TERM/I/intrusion_detection_system.html)

Wikipedia. (2005). Intrusion Detection System (http://en.wikipedia.org/wiki/Intrusion_detection_system)

Zanero, S., & Savaresi, S. M. (2004). Unsupervised Learning Techniques For An Intrusion Detection System. *ACM Symposium on Applied Computing*. March. Nicosia, Cyprus: ACM, 412–419.

Table 2.1 Snort Components and Functions

Snort Components	Function
Packet Decoder	Takes packets from different network interfaces and prepares the packets to be processed or to be sent to detection engine.
Preprocessors	To arrange or modify data packets before the detection engine does some operation to find out if the packet is being used by an intruder.
Detection Engine	To detect if any intrusion activity exists in a packet by employing Snort rules for this purpose.
Logging and Alerting System	To log activity or generate an alert depending upon what the detection engine finds inside a packet. Logs are kept in simple text files, tcp-dumpt style files or some other form.
Ouptut Modules	Control the type of output generated by the logging and alerting sysem

Table 2.2 Agent's Behavior

Agent's Behavior	Description
Social	Able to communicate to signal interest or information to other agents that constitute a part of its environment.
Autonomy	Able to operate on their own without the need for human guidance.
Reactivity	Able to perceive their environment and respond in a timely fashion to changes that may occur.
Adaptability	They are characterized by their flexibility, adaptation and facility to set up their own goals based on their implicit purpose.
Complexity / Intelligence	Able to behave as a result of their perception, knowledge and interactions.
Learning	Ability to learn from their experience and alter their future action sequences and behavior such that future mistakes can be avoided.
Pro-Activity	Ability to take the initiative rather than acting simply in response to their environment.
Goal-Oriented	Goal driven behavior that their action will cause beneficial changes to the environment.
Mobility	Ability to move from host to host.

Table 2.3 Definition of Terms

Aglet	Defines the fundamental methods for a mobile agent to control mobility and lifecycle.
Aglet Proxy	The interface to act as the agent representative or shield object that protects an agent from other malicious agents. Also provides location transparency and handles message passing.
Aglet Context	The stationary, uniform execution environment for the mobile agent, implementing security and management.
Message	The objects exchanged by mobile agents for the purpose of task collaboration and information sharing.

Table 2.4 Aglet's Characteristics

Aglet's characteristics	**Descriptions**
Mobility	Aglets provide a very simple Java API to implement the mobility feature.
Autonomy	Aglets can be programmed to make intelligent decisions, execute code and also move to and run on any machine that supports them.
Response Time	Aglets have rapid response time, they can visit several sites, negotiate with local software at each site and can return to their home base in order of a few seconds.
Concurrency	Multiple aglets with similar objectives can be dispatched simultaneously to accomplish various parts of a task in parallel.
Local Interaction	Mobile aglets can interact with local entities, such as databases, file servers and stationary aglets, through method invocation and with remote entities is by message passing.

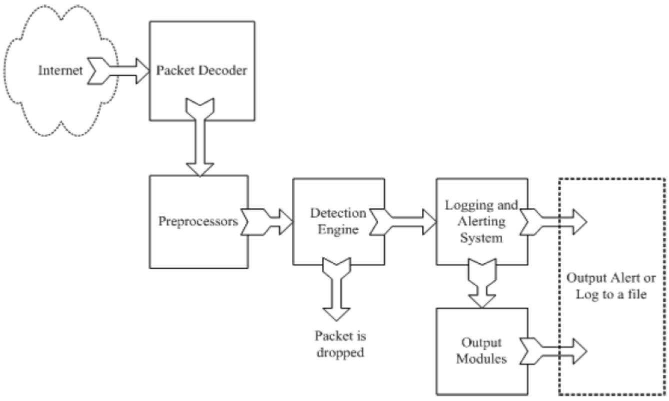

Figure 2.1 Interactions between Snort Components (Rafeeq Ur Rehman, 2003)

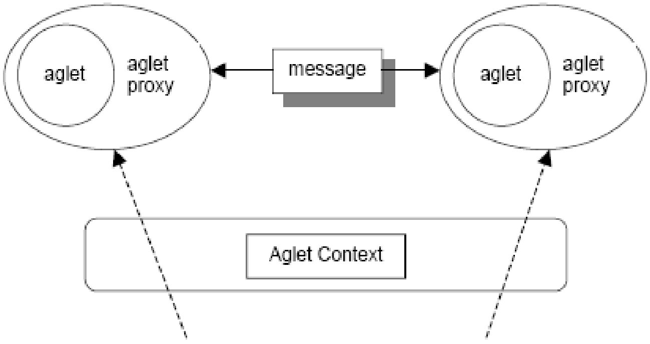

Figure 2.2 Major Interfaces and Classes in J-AAPI (AV Scott, 2000)

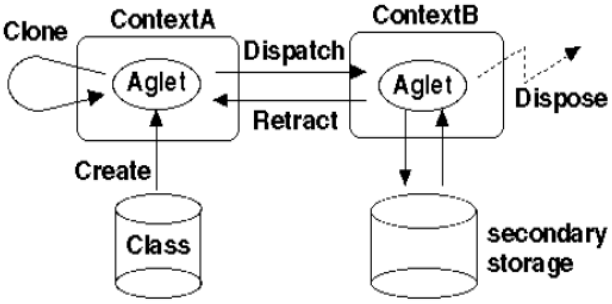

Figure 2.3 Aglet's Life Cycle (Mitsuru Oshima et al, 1998)

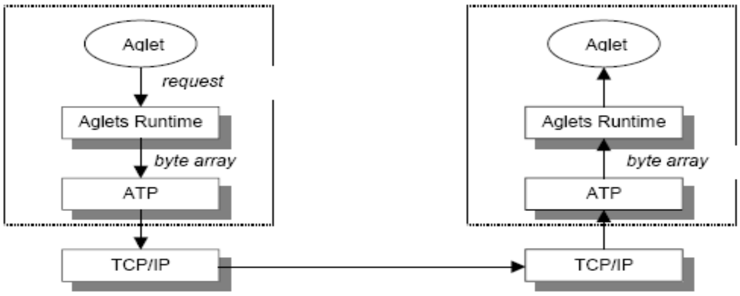

Figure 2.4 Aglet's Architecture (AV Scott, 2000)