APPLYING MOBILE AGENTS TECHNOLOGY TO INTRUSION DETECTION AND RESPONSE

CHEW WAI MENG
(B.Eng (Hons), Glasgow)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2003

Acknowledgments

I would like to send my gratitude to many people who have helped me as I worked on my thesis, and as I made my transition from Electronics Engineering to Computer Science. I would like to express my appreciation to Dr. Lam Kwok Yen for giving me the opportunity to embark on this area of research. Many thanks also go to A/Prof. Chi Chi Hung for his valuable advice and encouragement through the most difficult phase of my project. This thesis is able to be completed because A/Prof. Chi has given me inspirational leadership and guidance throughout.

I would like to thank my labmate, Mr. Li Tie Yan for numerous brainstorming and discussion sessions.

I would also like to thank the examiners for taking their precious time to review my thesis.

On the personal side, I am grateful to my wife and my mother for having faith in me and providing me the background motivation in all my life.

THANK YOU!

Contents

List of Figures

Abstract

As the capabilities of intrusion detection systems (IDSs) advance, attackers may disable organizations' IDSs before attempting to penetrate more valuable targets. To counter this threat, we present an IDS architecture that is resistant to flooding denial of service (DoS) attacks. The architecture frustrates attackers by making IDS components invisible to attackers' normal means of "seeing" in a network. Upon a successful attack, the architecture allows IDS components to relocate from attacked hosts to operational hosts thereby mitigating the attack. These capabilities are obtained by using mobile agent technology, utilizing network topology features, and by restricting the communication allowed between different types of IDS components.

Chapter 1
Introduction

1.1 Motivation

Over the last several decades, our society has rapidly become very dependent on computer technology. We have taken the controls for our whole civilization and loaded them onto digital machines. Our systems are responsible for storing sensitive medical information, guiding aircraft around the world, conducting nearly all financial transactions, planning food distribution and etc. A decade ago, the Internet was the refuge of researchers and academics. Now, as a major component of our population stares into computer screens and talks on mobile phones all day long for both business and personal uses, these technologies dominate our headlines and economy.

I am sure you have noticed that the underlying technologies behind computers and networks have many flaws. Sure, there are counterintuitive user interfaces and frequent computer crashes. Beyond these easily observed problems, there are some fundamental flaws in the design and implementation of the underlying operating systems, applications, and protocols. By undermining these flaws, an attacker can steal data, take over systems, or otherwise wreak havoc.

The concept of security is traditionally connected to the need of protecting confidential data from unauthorized access, but nowadays security is frequently approached from different perspectives. With the growing use of Internet infrastructure for commercial applications, modern systems tend to rely heavily upon networking and interoperation on public networks. As the Internet continues to grow, networked computer systems are more vulnerable to attack, and the number of attacks is growing exponentially.

In 1990, 252 incidents were reported to Computer Emergency Response Team (CERT). However, in just the first quarter of 2003, that number had grown to 42,586. In addition to the growth in the number of reported incidents, the number of systems involved per incident is growing – one recent incident involved several thousands of computer systems.[12] Furthermore, it seems probable that most incidents are not detected or reported.

Why are there so many attacks occurring? Today, the world of hacking is extremely large and difficult to categorize. However, several studies reveal computer attacks have similarities with many other crimes: - perpetrators who have many motives, including greed, revenge, the thrill of the chase, and peer pressure.[13] As the Internet continues to expand reaching billions of businesses and homes globally, online shopping is getting more popular.[15] Electronic commerce not only offers new services for customers but new opportunities for significant financial reward to intruders. It would seem likely that the problem will continue to worsen. Therefore, there is a need to find new security solutions and services.

In most cases, people that call themselves hackers create security breaches. In the early days of computer hacking, most of the hackers were hacking for self-projection. Hackers went professionals. Nowadays, intrusion is no longer the concern of computer intellectuals but instead has become the latest opportunity for criminal profit.[29] Many organizations have increasingly implemented various security systems such as firewall, IP traceback[40], digital certificates, VPNs (Virtual Private Network) and intrusion detection to combat system violations and security breaches. Perhaps the most promising among these is the use of Intrusion Detection System (IDS), IDS is widely deployed as a defense system because it can detect some set of intrusions and execute some predetermined action when an intrusion is detected.[4]

Today, intrusion detection encompasses the capabilities of event log analysis for insider threat detection, network traffic analysis, security configuration management and file integrity checking. There are several types of IDSs technologies: - misuse detection approach, anomaly detection approach, network detection mechanism, packet content signatures and etc. Another common categorization is between centralized system and distributed system. When an intruder launches an attack and breaks into a system, he or she will first be blocked by firewall using unauthorized access control mechanisms. However, there is always some security loophole that enables the attackers to bypass it and this is the time when IDSs play the important role to detect intrusions as soon as possible and alert the system administrators.

So far most research has been focused on developing the methods, improving efficiency and reducing the number of false positives (false alarms). Most of the existing IDSs have used central data analysis engines[24] that are arranged in a hierarchical structure, where the event information usually flow up to IDSs central analyzer and the actions are then relayed to the IDSs sensors. The monolithic architecture contains a number of problems that limit their configuration capacity, scalability of efficiency.[20,21] There has been concern over failure tolerance, as a monolithic system presents itself as a single point of failure and attack.

Due to the extensive use of IDSs, it has become a primary target for attackers. Web site operators are frustrated by the apparent inability of Internet service providers and Web host providers to quickly filter out denial of service (DoS) attack traffic when it pours into their routers and servers. Till now there is no silver bullet for DoS attack.

1.2 Analysis on Attacks

Threats from outside often are serious, of course. It would be a great mistake to underrate them or to write them off as some kind of media plot. It could be an even greater mistake, though, to let external threats distract you from the much greater vulnerability you face from inside your own organization. Below are two types of attacks:-

1.2.1 External Attacks

IP spoofing attacks: This is where the hacker steals an authorized IP address i.e. typically determining the IP address of a computer and waiting until there is no one using that computer, and then using the unused IP address. Spoofing is helpful for attackers who don't want to have their actions traced back, because the packets will appear to be coming from the system whose address the attacker is using. Additionally, IP address spoofing helps attackers undermine various applications, particularly those that dangerously rely only on IP addresses for authentication or filtering.

Packet sniffing: This is a common attack technique that gathers information from the local LAN, which could include userIDs, passwords, sensitive files or email. Passive sniffers gather traffic from the LAN without trying to manipulate the flow of data on the network. Active sniffing involves injecting traffic into the network to redirect packets to the sniffing machine.

Sequence number prediction attacks: Initially, in a TCP/IP connection, the two computers exchange a start-up packet which contains sequence numbers. These numbers are based on computer's system clock and then run in a predictable manner, which can be determined by the hacker.

1.2.2 Internal Attacks

Password attacks: Passwords are the most commonly used computer security tool in the world today. In many organizations, the lowly password often protects some of the most sensitive secrets. Unfortunately, with this central role in security, easily guessed passwords are often the weakest link in the security of our systems. A single weak password for one user on one account could give an attacker a toehold on a system. There are numerous freely available tools which can automatically guess passwords at extremely high rates, looking for weak password to enter a system.

Session hi-jacking attacks: This attack is based on a marriage of sniffing and spoofing. When a user has an established interactive login session with a machine using telnet rlogin, FTP, and so on, an attacker can use a session hijacking tool to steal the session from the user. When most hijack victims notice that their login sessions disappear, they often just assume it is network trouble. The users will likely just try to login again, unaware that their session was not dropped; it was stolen.

Shared library attacks: Many systems have an area of shared library files. These are called by applications when they are required (for input/output, networking, graphics and so on). A hacker may replace standard libraries for ones that have been tampered with, which allows the hacker to access system files and to change file privileges. A hacker might tamper with dynamic libraries. This would allow the hacker to possibly do damage to the local computer, send all communications to a remote computer, or even view everything that is viewed on the screen.

Technological vulnerability attacks: This normally involves attacking some part of the system (typically the operating system) which allows a hacker to access to the system. A typical one is for the user to gain access to a system and then run a program which reboots the system or slows it down by running a processor intensive program.

1.2.3 Denial of Service Attacks

As we have seen in the previous section, some attackers want to gain access to the systems, and use a variety of creative techniques to achieve this goal. Whilst other attackers are not looking to gain access; they want to prevent access by legitimate users or stop critical system processes. To accomplish this objective, they will utilize a variety of attack techniques to deny service. In the security community, such denial-of-service attacks are frequently referred to as “DoS” attacks.

Nowadays, many companies rely heavily on computer controlled systems, from environment control to factory robotics and automated warehouses. The disruption of these systems can shut down an entire business or be life threatening in the area of medical systems. A company that relies on electronic transactions for its livelihood could suffer serious financial damage if its systems are taken off line for even a short duration. There are incidents where an e-commerce company’s competitor launched a DoS attack against the company’s Web site, hoping that customers would abandon the target’s non-responsive servers and take their business to the attacker’s Web site.[35,28] According to [45], the total losses of US$123.7 million from information security breaches were reported by 163 organizations or about US$759,000 per organization.

Denial of Service (DoS) attacks is the most common and visible of all losses. While they often aren't technically elegant, DoS attacks can severely impact an organization, making defenses quite important. As shown in Figure 1-1, DoS attacks generally fall into two categories: stopping a service and resource exhaustion. Each of these categories of attack can be launched locally or across the network.

	STOPPING SERVICES	**EXHAUSTING RESOURCES**
LOCALLY	• Process killing • System reconfiguring • Process crashing	• Forking processes to fill the process table • Filling up the whole file system
REMOTELY (across the network)	• Malformed packet attacks (e.g Land, Teardrop, etc.)	• Packet floods (e.g SYN Flood, Smurf, and Distributed DoS)

Figure 1-1: Denial of Service attack categories

Stopping services locally prevents users from accessing them. An attacker could kill a process that provides the service, reconfigure the system to not offer the service, or even cause the service to crash. A logic bomb is a particularly nasty method for launching a local DoS attack. Another DoS technique is to locally exhaust resources. Attacks in this realm include filling up the process table, consuming the entire file system, or exhausting outgoing communications links.

An attacker could launch a DoS attack by remotely stopping services. A common technique for accomplishing this is to send a malformed packet that exploits a bug in the target operating systems or application, causing it to crash. The final category of DoS attacks is the most popular: remotely exhausting resources. In this type of attack, the adversary tries to suck up all available network capacity using a flood of packets. Several most popular techniques for launching a packet flood include SYN floods, Smurf attacks.

1.2.4 Distributed Denial of Service Attacks

A simple SYN flood allows an attacker to generate traffic from one machine. In a Distributed Denial of Service (DDoS) attack there are no inherent limitations in the number of machines that can be used to launch the attack and how much bandwidth the attacker can consume. DDoS represents a new and nasty turn in the evolution of DoS attacks, by allowing an attacker to coordinate the activities of an arbitrarily large number of hosts.[19]

To conduct a DDoS flood, the attacker will first take over a large number of victim machines, often referred to as 'zombies'. Potential zombie systems are located anywhere on the Internet and have a variety of simple vulnerabilities that the attacker can quickly exploit to take over the system. The attacker will scan large swaths of the Internet looking for vulnerable machines, exploit them, and install the zombie software on the systems. Most machines where zombies are installed are taken over using buffer overflow attack. Attackers will establish groups of hundreds, thousands, or even tens of thousands of zombies.

The attacker uses one or more client machines to tell all of the zombies to simultaneously execute a command, usually to conduct a DoS attack against the target. All zombies dutifully respond, flooding the victim in a bloodbath of packets. The client communicates with the zombies, but the attacker usually accesses the client from a separate system. This technique makes it more difficult for investigators to find the attacker. After finding zombies and locating client programs, the investigators still do not have the attacker, who is sitting at another machine, perhaps halfway around the world. The most popular DDoS tools, is the Tribe Flood Network 2000 (TFN2K), written by Mixter.

1.2.5 DDoS: A look at the future

DDoS pose an immense threat to the Internet; attackers constantly modify their tools to bypass the defense mechanisms. The move from a single or handful of machines launching a SYN flood against a victim to a coordinated attack from hundreds or thousands of systems represents a significant step in the evolution of attacks. This evolution and the future of DDoS tools is highlighted by Mixter,[27] the developer of TFN2K.

Currently, a great deal of work is being done in the computer underground to extend the concept of distributed attacks beyond TFN2K. One of those is the “stream” attack (discovered by Tim Yardley). Stream attack sends TCP packets with either ACK or both SYN and ACK flags set. Because they are not part of a connection, they will “confuse” a target machine and take some time to be processed by the operating system. If this attack is used in a distributed way, the attacker can overload machines with less hosts. It is very trivial to implement this feature. Another improvement is multicasting IP addresses. As multicast addresses are routed (forwarded) specifically by routers, they can multiply one packet into several ones. The concept would be to send out packets with a multicast (224.x.x.x) source. A target could send an error message back to multicast destinations, and multiply the bandwidth. Last but not least, attackers purposefully send special strings in the flood traffic, strings that Intrusion Detection Systems could falsely interpret as break-in attempts, the impact would be false alarms and the affected Intrusion Detection Systems could get overloaded or crashed.

1.3 Intrusion Detection System

An intrusion can be defined as any set of actions that attempt to compromise the integrity, confidentiality or availability of resource.[32] Intrusions are hard to catch because there are so many ways in which they may take place. In today's software development environment, the programming languages and operating system introduce a number of security flaws. These security flaws are difficult to detect and intruders are making use of these weaknesses to bypass existing security mechanisms.

In 1980, Anderson introduced the concept and terminology of intrusion detection. It has provided the early theoretical foundations for IDS. In his paper[18], he defined several terms and classified six categories of intrusive activities: - attempted break-ins, masquerade attacks, penetrations of security control system, leakage and denial of service attacks. In the beginning almost all intrusion detection systems were host based. Whilst in 1987, Denning extended Anderson's work through the introduction of generic detection model.[11] Besides focusing on generic model, she has provided a broad framework for future intrusion detection research. IDS are typically categorized into misuse detection approach and anomaly detection approach. Another common categorization is between network based approach and host based approach. In host based approach, we can also categorize them between centralized systems and distributed systems. Distributed intrusion detection systems have a number of advantages over their centralized counterparts such as scalability, graceful degradation of service and subversion resistance. In the following, we will comment on the advantages and disadvantages of centralized and distributed intrusion detection systems.

1.3.1 Centralized Intrusion Detection System

A centralized intrusion detection system is one where the analysis of the data is performed in a fixed number of locations, independent of how many hosts are being monitored. The event information from the sensors usually flows up whilst command and controls usually flow down. Figure 1-2, shows an event record of this type of architecture. The physical location of the event generators will be fixed since they monitor stationary resources. When all information is processed at a single location, the system is not scalable. The processing capacity of the analyzer unit limits the monitored network size and distributed data collection can lead to excessive data traffic over the network. Last but not least, a central analyzer is a single point of failure and a single target for an attack. If an attacker can disrupt such a failure point, a large portion of the network's IDS becomes inoperable. Current available intrusion detection system classified as centralized are: - IDES[12], IDIOT[38], and NSM[14].

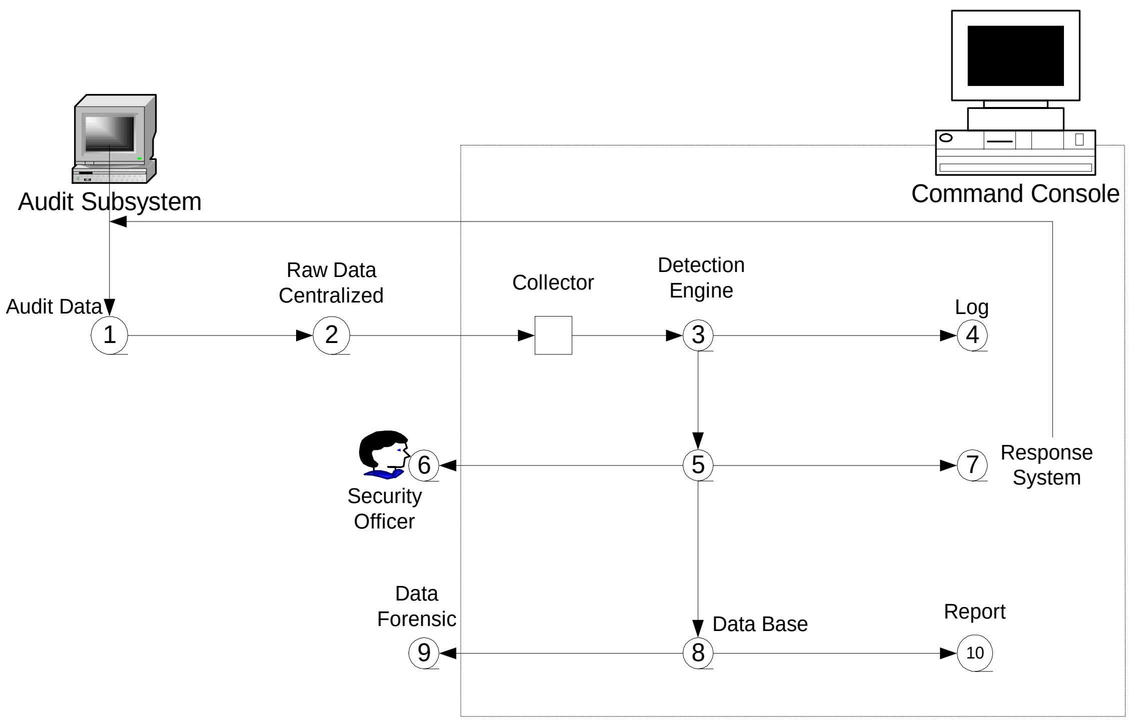

Figure 1-2: Centralized Intrusion Detection System

Typical events occurring in a centralized system (shown in Figure 1-2) are:-

1. An event record is created. This occurs when an action takes place, such as a file open or word processor. The record is written into a file that is usually protected by the operating system trusted computing base.
2. The target agents submit the file to the command console. This happens at predetermined time intervals over a secure communications link.
3. The detection engine, configured to match patterns of misuse, processes the file. Data records are parsed in their raw, or original format.
4. A log is created that acts as a data archive for all the raw data used in a prosecution.
5. When a suspicious activity occurred, an alert is generated. When a predefined pattern is observed, such as access to a mission critical file, an alert is

forwarded to a number of different subsystems for notification, response and storage.

6. The security officer is notified. This can be done through audible or visual methods; pager, email, SNMP.
7. A response is generated. The response subsystem matches alerts to predefined responses or can take direction from the security officer to execute a response. Responses include actions as reconfiguring the system, shutting down a target etc.
8. The alert is stored. The storage is usually a relational database. Some systems store statistical data as well as alerts.
9. The raw data is moved to raw data archive. This archive is rolled over periodically to reduce the amount of disk space used.
10. Reports are generated.
11. Data forensic is used to look for long term trends. The behaviour is analyzed using both the stored data in the database and the raw event log archive, where data from in-band and out-of-band sources may be correlated to detect a wide range of misuse.

1.3.2 Distributed Intrusion Detection System

A distributed intrusion detection system is one where the analysis of the data is performed in a number of locations proportional to the number of hosts that are being monitored. The event record of the lifecycle in Figure 1-3, is as follows:-

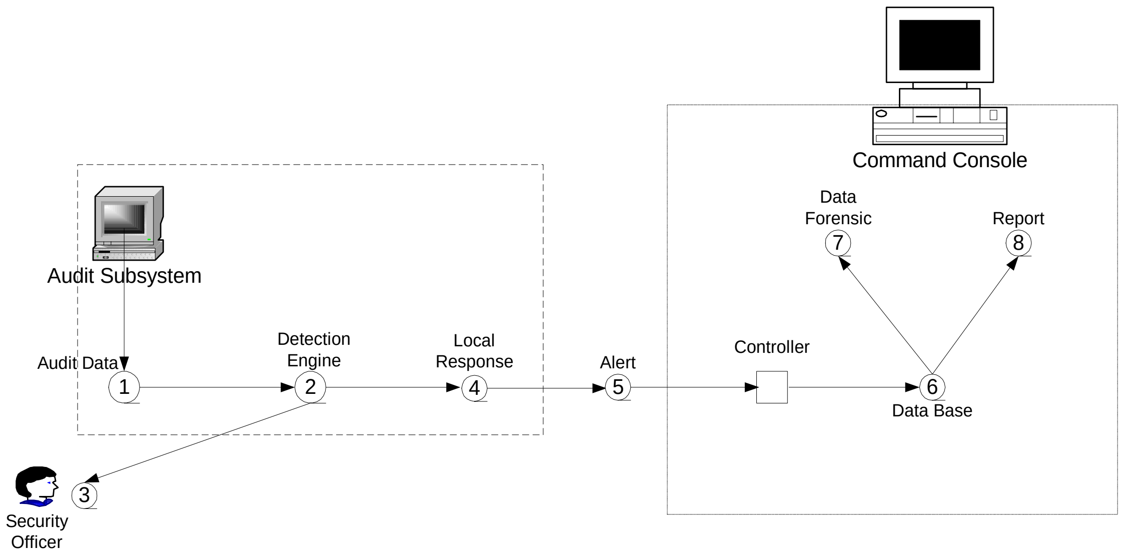

Figure 1-3: Distributed Intrusion Detection System

1. An event record is created.
2. The file is read in real-time and processed through a target resident detection engine. The range of detection is limited to a single target in this architecture.
3. The security officer is notified. Some system notify directly from target whilst others notify from the central console.
4. A response is generated. The response may be generated from the target or console depending on the architecture.
5. An alert is generated and sent to a central console.
6. The alert is stored. Statistical behavioral data outside alert data are not usually available in this architecture.
7. Data forensics is used to look for long-term trends.
8. Reports are generated.

The classic solution to combat the shortcomings of central analyzer is the introduction of several hierarchical layers and redundant components. One of the earliest distributed intrusion detection system (DIDS) is a joint project between UC Davis, Lawrence Livermore National Laboratory and US Air Force.[39] Most DIDS such as Cisco Netranger[8] are using hierarchical structure as shown in Figure 1-4. The hierarchical architecture involves four levels, the bottom layer is the IDS hosts that contain either host based or network based IDS sensors. At the lower layer, the IDS controllers are performing data reduction or aggregation. This is the advantage of subversion resistance, with the controllers cross checking each other. At the immediate layer, there are analysis controllers which receive the data transferred from lower layer and analyze them. They also communicate with the higher layer to report their analysis results. In higher layer, the decision making controllers will process the results and generate a report for the system administrator. The administrator can then manually assess status and issue the relevant commands.

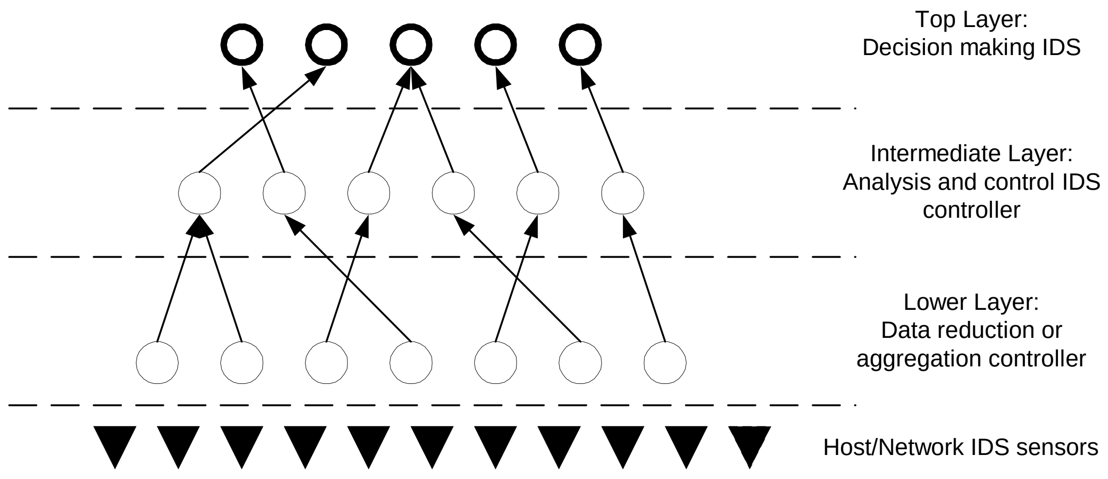

Figure 1-4: Hierarchical Distributed Intrusion Detection Architecture

1.4 Contributions to the Research

Many people have benefited from deploying automated IDSs within an organization's security architecture. Oddly enough, one of the most obvious benefits of deploying IDSs is also one of the current main drawbacks. The amount of data collected and notifications generated by current IDSs may quickly overwhelm most organization's security operations, especially if the systems are deployed without any customization based on the specific requirements of the monitored environment. With the new DDoS features being developed, IDSs may become a primary target for attackers. In order to prevent such threat, I have proposed an architecture that makes use of mobile agent technology that can evade further damage caused by flooding DoS attacks. I have published this idea in the *Proceedings of IEEE IPDPS 2002*.[42]

Agents, intelligent agents and agent based systems have attracted considerable interest from many fields of computer science. Agent technology has been academically applied in a variety of fields, particularly in artificial intelligence, distributed systems, software engineering and electronic commerce. In this proposed model, I have adopted the use of mobile agents and a combination of techniques. Firstly, critical IDS components are made adaptive to flooding DoS attacks in that they can be automatically relocate and backup in the event of an attack. Secondly, we make use of both static and mobile agents as building blocks. Various agents perform tasks in control, detection, and policy. Proxy agent group is introduced to frustrate the attackers. Suspected packets are also blocked before they can penetrate valuable targets in IDS. This IDSs model is not primarily a mobile agent model but it does make extensive use of mobile agent technology. This technology enables load balancing and provides backup capability.

Many people avoid mobile agent technologies because they believe them to be insecure. In order to prevent sensitive data from unauthorized hosts accessing and being tampered by malicious agents, we proposed an efficient key assignment scheme based on RSA to enhance the performance of Volker and Mehradad's scheme.[36] As mobile technology is implemented as a solution to protect IDS architecture; they are vulnerable to attacks and tampering. In order to prevent these attacks, mobile agents must follow some proper security policies or techniques such as access control, authenticating, credentials, code verification.[37] All agents arriving at the hosts are authenticated by their unique identification. This authentication scheme is used all over this architecture. It can be used to check if the agents stay in their proper regions.

1.5 Thesis Organization

Chapter 2 of this thesis surveys some related works on the weakness of current intrusion detection system. We will highlight the Distributed Denial of Service attack that will disable IDS architecture and discuss how mobile agent technology can be the solution to this problem.

Next, Chapter 3 presents our approach for protecting IDS based on mobile agent technology. In this chapter, we will examine how the system prevents penetration attacks from disabling IDSs.

Chapter 4 provides a more detailed consideration of protecting mobile agents using efficient authentication scheme.

Finally, Chapter 5 discusses what conclusions can be drawn from the work, the drawback in our solution and possible improvements.

Chapter 2

Related Works

2.1 Mobile Agent Techniques

Mobile agents offer several potential advantages when used in IDSs that may overcome limitations that exist in IDSs that only employ static, centralized components. What is mobile agent? A software agent is a piece of code that can run on host, perform transparent migration to another host, and resume its running state. The agent comprises of code and state information needed to carry out some computation and requires an agent platform to provide the computational environment in which it operates. Agents may be static or mobile. Stationary agents remain resident at a single platform, whilst mobile agents are capable of moving from one platform to another and interact with each other. In order to accomplish their task, mobile agents can also gather data and use services present on visited hosts. The mobile agent characteristics could be addressed as follows: -

- the rapidity of execution due to the small quantity of code the mobile agents represent. This is particularly desirable to respond to the attack as soon as possible.
- reducing network load. Instead of sending huge amounts of data to the data processing unit, it moves the processing algorithm (i.e agent) to the data.

- the ability to adjust their execution code depending on the characteristics of the machine they visit. This factor is also important since it enables the mobile agents to adjust the defense parameters to better protect the system.
- the mobility which is the main propriety. As mobile agents can travel across the network they can filter the relevant information from different machines. They have the ability to correlate all this information and adapt the answer. For instance it could be helpful if one attack is coming from several sources or if one attack reaches several destinations. They can also use these migration abilities in order to limit the possibilities of interaction between the agents themselves and a potential offensive piece of code.

It is obvious that applying mobile agent technology into intrusion detection system will be more encouraging if we can combine the full intrusion detection capabilities with the mobility property of agents. This combination may need to separate the IDS into many small pieces of functional units--- the IDS components. These components are wrapped into mobile agents to make them the IDS agents. Therefore, the combined system is actually built on the underlying mobile agent paradigm. In the research area, one of the successful IDSs of using agents--- AAFID, is a very typical approach of using autonomous agents for intrusion detection.

2.2 MAIDS Approaches

In the literature, many researchers have conducted the agent based intrusion detection system solving different problems. Although some of their systems have been studied for years, no ideal result is published and no completed project is applied well into practical system. IDSs are still in their infancy. They are:

1), CERIAS at Purdue University developed a distributed IDS called Autonomous Agents For Intrusion Detection.[2] AAFID was the first architecture of using autonomous agents for intrusion detection. The system is based on independent entities called autonomous agents for performing distributed data collection and analysis. The agents report to Transceivers on a per-host basis and the transceivers further report to the higher level Monitors that have control and data processing role. Their hierarchical architecture allows data to be collected from multiple sources, thus being able to combine the best characteristics of traditional host based and network based IDSs. The modular characteristics of the architecture allow it to be easily extended, configured and modified.

2), Helmer et. al at Iowa State University developed a system of using mobile agent technologies and collaborative information to implement a prototype IDS: "intelligent agents for intrusion detection".[17] This system is a layered system of using data mining techniques for detecting intrusions. The agents at different layers perform different parts of the data mining procedure such as data cleaning agents, data classifying agents and data mining agents. This system was more focused on implementing the agent's internal intelligence than on using the agent's mobility feature.

3), JAM project conducted at Columbia University[44] was a distributed data mining approach. The system has two key technologies: 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. Agents were used here mainly for sharing knowledge (meta-learning) from different remote classifiers.

4), Other approaches include Intrusion Detection Agent (IDA) system[22] in Japan and Intrusion Detection System based on Mobile Agent (IDSMA)[23] in Brazil. The IDA system used two kinds of agents, the information gathering agents and tracing agents, for collecting information and tracing intruders in a local area network. While IDA may be suitable for LAN, the system design must be reconsidered to fit large-scale network. IDSMA presents a hierarchical architecture for using mobile agent in IDS. It uses a large number of small mobile agents to perform all the tasks of monitoring, decision-making, notification and reaction to attempted intrusions. The authors claimed a clear layered model as the framework and implemented part of the functions. However, we can only evaluate the system given more detailed design information.

From the above survey, we see that the usage of mobile agent technology does help to build a better hierarchical IDS with many precious properties like: continuous autonomous running, fault tolerance, scalable and adaptable, they still suffer from a major problem that an un-secure mobile agent platform may even shutdown the IDS. Specially, we introduce a DDoS attack against MAIDS in the following section.

2.3 IDS suffered from DDoS Attacks

2.3.1 Distributed Intrusion Detection System

The powerful Distributed DoS attack tools are like Tribe Flood Network 2000 (TFN2K) and Stacheldraht.[5] These attacks typically exhaust link bandwidth, router processing capacity, to achieve the objective of breaking network connectivity of the victims. One of the most interesting features of TFN2K involves the communication between client and zombies. In order to prevent other attackers or the zombie machine's administrator from accessing the zombie, the client must authenticate to the zombies using an encrypted password. Then all the packets from the client to the zombies are sent using an ICMP Echo Reply packet. TFN2K communicates using a ping response, without ever sending a ping. First, ICMP Echo Replies are allowed into many networks, because the network administrator configures routers and firewalls to allow inside users to ping the outside world. Their ping responses have to get back in, so ICMP Echo Reply packets are allowed. Another reason for using ICMP is to make the connection more stealthy. There is no port number associated with ICMP; the system just listens for ICMP packets and passes them to the TFN2K application. Therefore, if the administrator runs Nmap to conduct a port scan of the zombie machine or runs the *netstat –na* command locally to get a list of open ports, no new ports will be listed as open for TFN2k, because it uses ICMP.

TFN2K communication also supports a variety of stealth mechanisms. First, the source address of all traffic from the client to the zombies can be spoofed. Further, the zombies themselves spoof the traffic sent to the victim machines. The servers can even send out decoy packets to other victims to help throw off an investigation. When an investigation into a DDoS attack occurs, the end victim has to trace the attack back, router by router, ISP by ISP, to one or more of the zombies. From that point, the attack must be traced back, again router by router, ISP by ISP, to the client. Even then, we have not yet found the attacker, who is connected to the client using Netcat, possibly forwarded along a Netcat relay network. In other words, finding the attacker with a truly robust TFN2K deployment is very difficult.

DDoS attack is particularly damaging. After gaining access to the target systems, most attackers want to ensure that other intruders will be kept off from the system. The more experienced attackers will harden the system, installing security patches and shutting down irrelevant services to prevent other attackers from gaining access to the system. Next, the attackers want to maintain that access. In order to keep access and control of the systems, attackers utilize techniques based on malicious code such as Trojan horses, backdoors, and RootKits.

Netcat is one of the most popular backdoor tools in use today.[1] Firstly, the attackers compile it with its "GAPING_SECURITY_HOLE" option, so that Netcat can be used to start running another program on the victim machine. After loading the Netcat executable onto the victim machine, Netcat will listen on TCP port 12345. When the attacker connects to TCP port 12345 using Netcat as a client, the Netcat backdoor will execute a command shell. The attacker then has an interactive shell session across the network to execute any commands on the victim machine. The context of the shell session will be the same as the attacker when she or he executed the Netcat listener. A backdoor, ideally will continue to provide access for the attacker even as the system configuration changes, with users being added and deleted. Attackers understand that backdoor utilities must have names that will not attract any undue attention. A properly constructed backdoor will still be usable by the attacker to gain access even if the original entry point is closed by a system administrator.

Upon determining the location of critical IDS components, the malicious code opens a channel for the attacker to launch a flooding DoS attack. Even if an organization became aware of the reconnaissance code, by the time a response is initiated, the attacker would have gained a view of the organization's internal IDS topology. Upon discovery of IDS topology, the attacker would like to penetrate and control the distributed IDS. However, critical IDS components are likely to be well maintained and difficult to penetrate but the malicious code can eventually increase the rate of attack. Without the critical aggregation, analysis and reporting capabilities, the IDS will not be able to effectively detect and respond to attacks.

2.3.2 Existing solutions against DDoS Attacks

The seriousness of the Distributed DoS problem and the increased frequency of DDoS attack have led to the advent of numerous defense mechanisms. However these solutions have some drawbacks. Most of the mechanism require certain features to achieve their peak performance, and will perform quite differently if deployed in an environment where these requirements are not met.

Most zombies are deployed by attackers using standard exploits against unpatched systems; one must keep the systems patched and up to date. However, because some attackers may still break into the systems and install a zombie, another solution is the filtering mechanisms which filter out attack streams completely.[25] Examples include dynamically deployed firewalls and also a commercial system TrafficMaster. As DDoS attack always involve spoofed packets, egress anti-spoof filters will be useful in protecting zombie running on one of the machines. These filters will drop all outgoing traffic from your network that does not have a source IP address found on your network. However, it runs the risk of accidentally denying service to legitimate traffic.

There are other several countering solutions against DDoS attack, such as installing extremely fast computers, have adequate bandwidth, have redundant paths through multiple ISPs. Still, even with all these mentioned solutions that an organization can afford, a large enough grouping of zombies can easily overwhelm any network. In reality most organizations simply cannot afford to buy bigger bandwidth to handle massive DDoS attack.

2.4 Intrusion Detection System Evasion

IDS evasion is a very active area of research in the computer underground right now. New tools and techniques are being devised to avoid IDS, and existing techniques are being added to older tools.

2.4.1 IDS Evasion at the Network Layer

IP offers the ability for the network devices to fragment packets to optimize the packet length for various transmission media. A large IP packet is broken down into a series of fragments, each with its own IP header. The fragments are sent one by one across the network, where they are reassembled by the destination host.

When these fragments pass by network-based IDS, all of them must be captured, remembered, and analyzed by the IDS. A large number of disparate fragment streams, spread out over a long time, means that the IDS must have considerable long-term buffers to store all of this data. Therefore, IDS require a great deal of memory and processing power to gather and analyze fragments. Furthermore, to analyze the communication reflected in the fragments, the IDS must reassemble all of these packets in the same way that the target system does reassembly.

Let’s explore an example of how an attacker may fragment packets to evade IDSs detection. The “tiny fragment” attack is designed to fool the IDS by creating an initial fragment that is very small. The packet is sliced in the middle of the TCP header. The first fragment is so small, in fact, that it does not contain the TCP port number. Instead, the TCP port number follows in the second packet. Suppose the IDS is looking for traffic on a specific port, such as TCP port 23, to warn administrators when someone tries to telnet but because the IDS is looking for the port number to make filtering decisions, it may ignore the tiny initial fragment as it passes. After all, the first fragment does not have a port number in it. Also, the IDS may allow the second fragment without a concern. After all, it's just part of the original packet associated with the first fragment. In this way, the attacker has managed to send in two packets that avoid detection by the IDS.

2.4.2 IDS Evasion from DDoS Attacks

A lot of previous works has focused on detecting DDoS attacks and mitigating their detrimental impact upon the victim.[9] This approach does not eliminate the problem, nor does it deter potential attackers. Given the damage that can be inflicted through DDoS attack, the best defense against a massive DDoS attack involves rapid detection and the ability to response efficiently. Therefore, we need to employ IDS tools that can quickly alert you when a DDoS attack starts.

Our approach was inspired by some early works done by [7, 16, 17, 31, and 34]. These works includes implementing lightweight agents for intrusion detection, using mobile agents to counter DDoS attack and thwarting attackers by hiding critical IDS components. In [7], researchers have developed a framework named Sparta (which is an acronym for Security Policy Adaptation Reinforced Through Agents), which heavily relies on mobile agents. The goal of Sparta is to design a mobile agent based IDS that identifies and improves potential shortcomings of other intrusion detection system designs. In the design, each host has at least a local event generator, storage component and the mobile agent platform installed. Agents can be seen as guards, which protect a network by moving from host to host and performing random sampling. Instead of monitoring each host at any time, agents only visit machines from time to time to conduct their examination.

In Peter Mell's design,[31] he proposed the use of mobile agent technologies to seamlessly relocate critical IDS components from attacked hosts to hosts that are still operational. Thus, the IDS components become invisible to an attacker's normal means of seeing in a network, such as passive sniffing, active network monitoring, and host penetration and analysis. The IDS components become invisible by using assumptions about the network topology and by restricting the communication allowed between certain types of components. In the event that a critical component is attacked, then the component moves to an operational host. When it may appear impossible for an agent to move from an attacked host, we use mobile agent technology to enable a type of backup system for processes. Thus, the agents on attacked hosts can become disabled and mobile agents on other hosts will automatically pick up the disabled components' duties.

Another challenging issue when building IDS using mobile agents is how to relate information from different sources. How can there be cooperation and communication between agents themselves? Helmer[16] has suggested using lightweight agents to do event correlation. The proposed design includes: - (1) static data cleaning agents that obtain information from system logs, audit data, (2) low level agents that monitor and classify ongoing activities, (3) facets for the low level agents that add cooperation to the agents, (4) data mining agents that use machine learning to acquire predictive rules for intrusion detection from system logs. The agents themselves communicate directly only to their related data gathering agents and mediators. This will allow agents to fuse related data in real time and take advantage of knowledge about the security status of related components in the system.

In the next chapter, we will further study the evasion solution of using mobile agent based architecture. We design the secure architecture for protecting the critical IDS components and through the backup and flow control mechanisms, the system will be proved to be secured against DDoS attacks.

Chapter 3

Intrusion Response Using Mobile Agents Technology

3.1 The Limitations on current design

In the last chapter, we pointed out the problems faced by centralized IDSs and how DDoS attack can freeze or shut them down. Therefore, we adopted Peter Mell's design and improved on it. The essence of his approach is the implementation of proxy region with proxy agents. However, there are too many restrictions and assumptions that limit its usefulness in the real IDSs.

Firstly, one of the inequitable assumptions made in Peter Mell's design is that the network backbone including critical and proxy hosts is not penetrable. It could be true that the critical hosts are well configured and are not penetrable via network attacks. However, it is inappropriate to claim that proxy hosts are not penetrable. According to their definition in the design, all of the intermediate elements in the network are included into proxy region. According to Peter Mell's assumptions, the IDSs' applications are built such that an attacker can exploit no flaws in gaining unauthorized access. However, according to [33], the failure of the system is mainly due to the weak security in the modern complex software.

Secondly, all of the child hosts are resided in the regions (usually contain hosts and servers used by an organization). These child hosts are not allowed to initiate connection to any other regions. If the network is only used for intrusion detection approach, this assumption would be feasible. But most networks will be used for some sort of applications and therefore they must have communications between network elements.

Last but not least, the ignored problem of their proposal lies in the central directory server problem. In Peter Mell's design, mobile agent technology was being used to secure the IDS architecture. In the event when an attacker cannot locate the critical IDS hosts/agents, the next target will be on the mobile agent directory server. To solve the above mentioned problems, we have proposed the well-known and widely used RSA public key cryptosystem. The agent code is signed and can be authenticated before it is executed (to protect the platform).

3.2 Background concerning our model

To counter the threat of attackers finding and disabling IDS components, we have proposed a model using passive response system. It is an anticipated way to place the system on the defensive without disturbing too much of its operation. Instead of actively trying to stop an attacker's actions, our proposed model attempts to hide IDS components and move them away from harm. Thus, our IDS components become invisible to an attacker's means of seeing in a network: passive sniffing, active network monitoring, and host penetration. In the event that a critical component is attacked, then the component moves to an operational host. Whilst it may appear impossible for an agent to move from an attacked host, we use mobile agent technology to enable a type of backup system for processes.

Applying software agents to intrusion detection is not entirely new. One noteworthy DIDS is Autonomous Agents for Intrusion Detection (AAFID) developed by Purdue University. AAFID is in many ways a classical DIDS with agents used mainly as a means for structuring the intrusion detection collection component into a set of lightweight software components. In our proposed model, we have enhanced the performance of Peter Mell's architecture, by making several changes. Firstly, we do not restrict the communication flow between different regions since they may need to cooperate with one another. Secondly, we didn't include the proxy region (agent) into the backbone region since there are many proxy agents in operation and to make all impenetrable is impossible. A backbone is a set of network elements that are typically secure against penetration from attackers on the network: firewalls, routers, and switches. Backbones are also allowed to contain security devices that are secured against penetration from network attacks. Thirdly, we removed the central directory server with several region based servers at different layers.

In Figure 3-1, we defined the network into several domains: Domain A, B and C. Every domain contains special security hosts that are mobile agent enabled. The security hosts consists of critical, proxy and leaf. These domains can communicate securely with IPv6 with IPsec protection. IPsec stipulates a mandatory authentication protection for "IP Header" and an optional confidentiality protection for the endpoint-identity information which is in some "IP Header Fields". As IPsec is offered at the IP layer, any higher-layer protocol such as TCP and UDP can take advantage of IPsec capabilities. By implementing IPsec protection, Domain A has created a "secure path" with Domain B, or between Domain C, creating so called virtual private networks (VPNs).

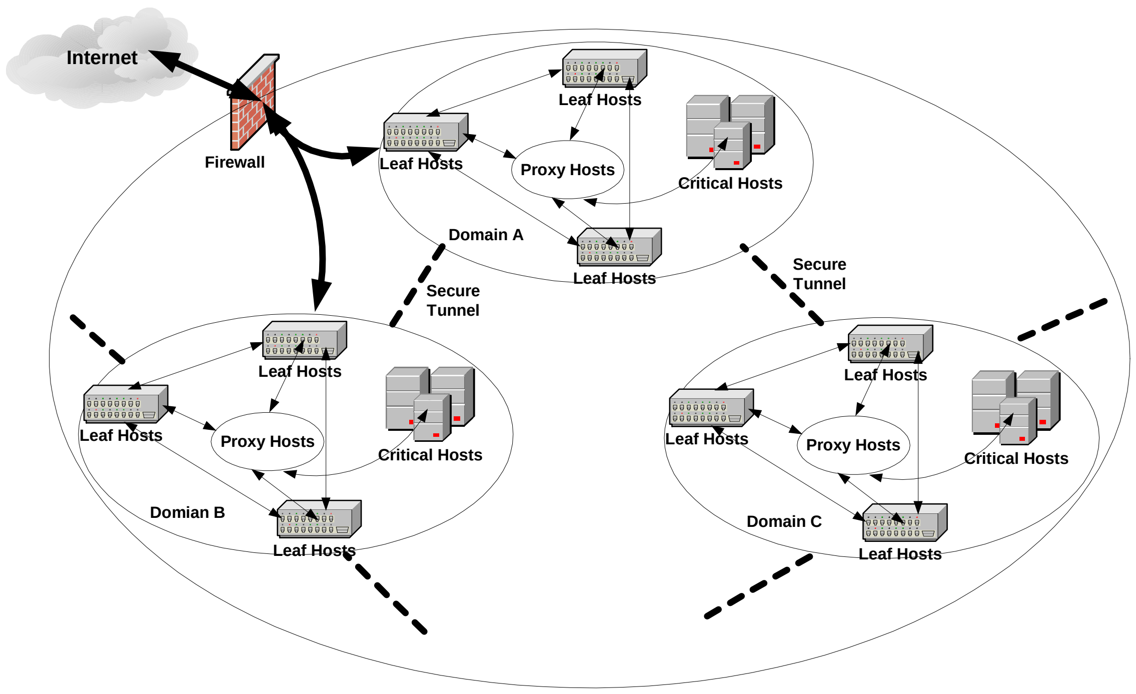

Figure 3-1: The Enterprise network and domains

3.3 Description of Security Hosts and Placement within a domain

3.3.1 Agents Involved

Critical region composes of the critical hosts that may include the important application server and also the critical IDSs hosts. Critical hosts are residing in this region and house the critical agents to perform intrusion detection aggregation, analysis and control. The critical agents are the most important to protect against attacks. Similar to many existing IDSs, if the packets are not from an authorized source, the critical host quietly drops the packets without sending any reply. This region can be any network topology but its network bandwidth must be wide enough to ensure the internal high-speed communication. The communication between this region and proxy region must be wide too. Therefore, the communication between these two regions cannot be easily flooded by attacks. The critical servers are also responsible for connecting to the proxy group coordinators.

Proxy region is composed of all the intermediate layer hosts or networks elements. Proxy region also house the proxy agent responsible for receiving (incoming) packets from and sending (outgoing) packets to the leaf agents. Critical hosts are not allowed to directly communicate with the leaf region. Instead, critical and leaf region need to communicate through proxy region. The controller agent can be found inside this region. The controller agent is responsible for load balancing. When the host is overloaded, it will notify the leaf agent not to send any more packets of new sessions to it.

Leaf region is usually the local area network (LAN) in the network. It comprises of working machines/servers used by organizations and IDS components such as the host IDS's sensors. Gateway agent is an agent that resides in this region. It is responsible for grabbing packets from the external network and sending them to one of the controller agents in proxy region. The mobile agent that works in this region cannot move to another region. However, we didn't restrict the communication between two leaf regions because they may be two cooperated departments of a company.

3.3.2 Identifying Attacks by Agents

When attack events occur, the agents must be able to discover the scenario as an intrusion. There are three possible ways to describe attacks. Firstly, it is to implicitly describe attacks by providing code that directly operates on data structures delivered by data gathering components. The code itself determines whether an intrusion has occurred by processing the input and calling appropriate response functions.

Another possible way that separates ID systems into components is the specification of scenarios in an application-specific scripting language. Usually, one is supported by predefined data types (e.g IP packets) or a rudimentary way of expressing timing constraints.

The last approach is a special language which allows the security officer to define attack patterns which consist of a set of events that can be spatially and temporally related. The description of the attack is translated into rules and code, which can directly be processed by agents. This has the advantage of an intuitive description of the attack scenario.

3.4 The Organization of Proxy Region

As shown in Figure 3-2, cluster G is a multicast group and all the proxy agents are members of the cluster. Within the cluster, agent can share information of detection and intrusion. Each of the members in group G has a shadow in the mirror cluster G'. For example, A_1 and A_1', A_2 and A_2', C and C' are all "buddy" agents. As the buddy agent group G' is not a multicast group so that there is no group communication inside G' (this is to permit only one to one communication). Agent C and C' are their group coordinators respectively. The main objective of such a structure is that we want the proxy agents to be protected by their "buddy" agents. The structure can also help to remove the central directory server by reporting to a group of different agents.

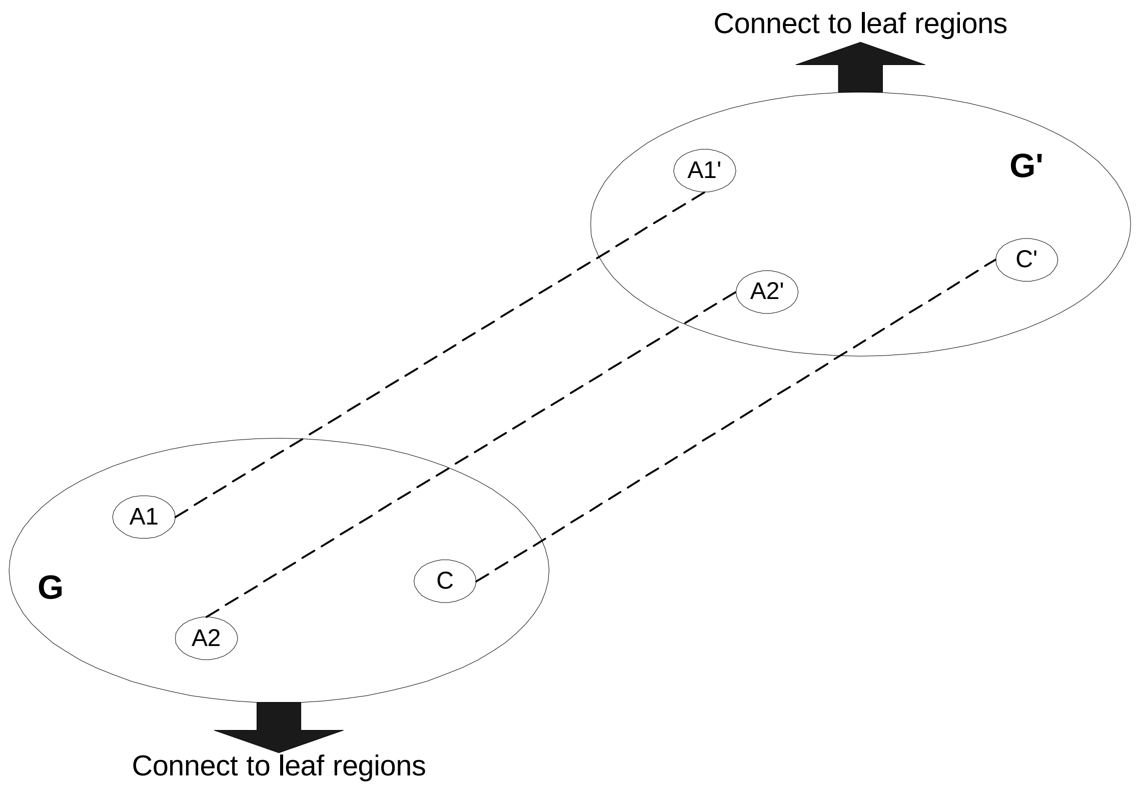

Figure 3-2: The schematic of proxy agents group

Note that A and A' are the "buddy" agents, the same goes for C and C'. The buddy agent will be generated at the same time and distributed with a pair of effective access keys. We will discuss the cryptographic keys assigning in mobile agent environments in Chapter 4. If one agent is in danger, the other agent can backup its ruined "buddy" agent by the backup mechanism. The proxy region is divided into several different network segments. We have carefully arranged the proxy agents in different network segments. In the event of penetrated host, its agents cannot be used by the attacker as the sniffing platform for attacking agents on other segments.

3.5 Backing up mechanism

In the event that an attacker eliminates an agent, the backup is very important for the protection of the agent system. It has to be resumed quickly and completely so that it would not disrupt IDSs detection and response. A simple voting scheme is used to decide which agents should be the successor. The backup agents maintain full or partial state information of the agent they are backing up. The resurrected agent will then be able to resume to its full or partial state of functionalities. Every agent will have one or more backup in the event of attacks. However, the tasks of backup mechanisms for agents in the three regions are different.

3.5.1 In the Critical Region

There are two backup agents for agents in critical region. When a critical agent is down, these two backup agents will negotiate a successor and this elected one will resume as the critical agent. It will then negotiate for its own backups. The backup agents can only stay inside the critical region.

3.5.2 In the Proxy Region

The backup mechanism for the proxy agents is a little more complex. In Figure 3-3, A and A’ are two “buddy” agents. B and B’ are the backups of A and A’ respectively. We let A keep contact with B’ and A’ keep contact with B for the reason that if A is ruined, A’ can help to resurrect A with B and vice versa. Agent A is not allowed to keep contact with its backup B because if A is ruined, B will fail too. We designed this scheme because we can use the relationship of A and A’ (they are “buddy” agents and can coordinate with each other). Therefore, whenever one of the agents is under attack, the other will quickly detect it and resume as a new one.

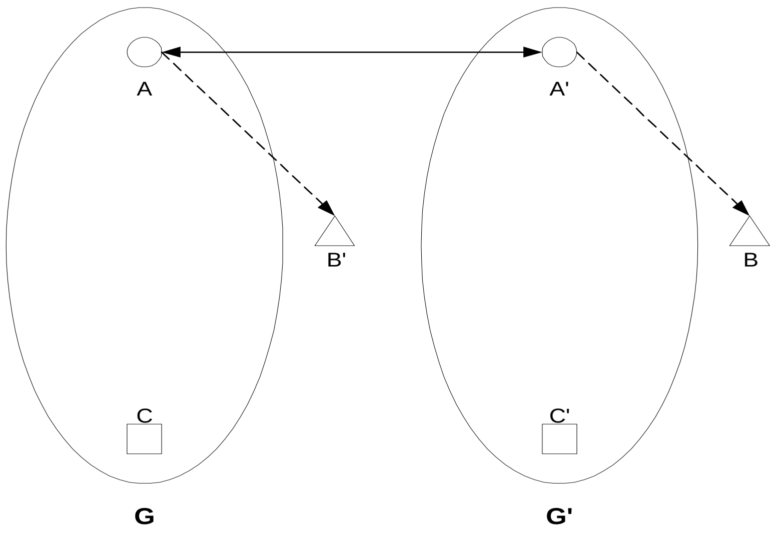

Figure 3-3: The backup of proxy agents

At normal status, A’ will initiate connections to A on a fixed time interval. In one connection, they will exchange their locations and current status. In the event when A’ detects a wrong status in A, A’ will decide that A is being ruined and it will soon replace A with agent B. As A’ always keeps the latest location of B and frequently updates A’ current status to B. Furthermore, A’ will backup itself once and let the resurrected agent A keep contact with it. The original backup for A’, B’ will lose as A is ruined. B’ will disable itself if it cannot receive any responses from A for a specified period. The backup mechanism can only work with the help from their group coordinators C and C’ that provide their current locations to each other. Also the backup mechanism for C and C’ is almost the same, but it needs help form the critical server which keeps their current status. In this scheme, all of the proxy agents and their backup are situated in the proxy region.

3.5.3 In the Leaf Region

The leaf agents will not keep one or two backups of themselves. Therefore, if a leaf agent is not functioning, its responsible proxy agent will detect it. The proxy agent will then seek its buddy agent for the current location and status. Then it creates a new one. However, the attacker who disabled this leaf agent may still be in the leaf region and continuing to sniff the network traffic. If the attackers discover the resumed agent, he or she will disable it immediately. In this case, we have to repair the network to find out the intruders first to prevent further attacks. Therefore, we locate the backup agent on different hosts.

3.6 Backing up critical agents

From the above description, critical agents can only communicate with the leaf agents through proxy agents. To secure the critical region, we do not allow the connection initiated from other regions to the critical region, so that a proxy agent need not know the current location of the critical agents. Therefore, a connection between a critical agent and a proxy agent can only be initiated by the critical agent. After initialization, critical agent and proxy agent can start to transfer data or command when the connection is up.

This applies for the connection between a proxy agent and a leaf agent being initiated only from a proxy agent. A leaf agent can report its request or current location to group multicast address. Hence, it will not know its responsible proxy agent's location until proxy agent connects it. This top-down security structure makes it very hard for the attacker to start an attack from the bottom level. Also the protection of the agents is also from top level to bottom level.

In Figure 3-4, we can see that in the critical region, the critical agents can protect themselves by backup. They can also protect the group coordinator of C and C'. Furthermore, C and C' can protect their group members (A and A' respectively). The "buddy" agents in group G and G' can protect each other. A and A' will then protect the leaf agents they are responsible for.

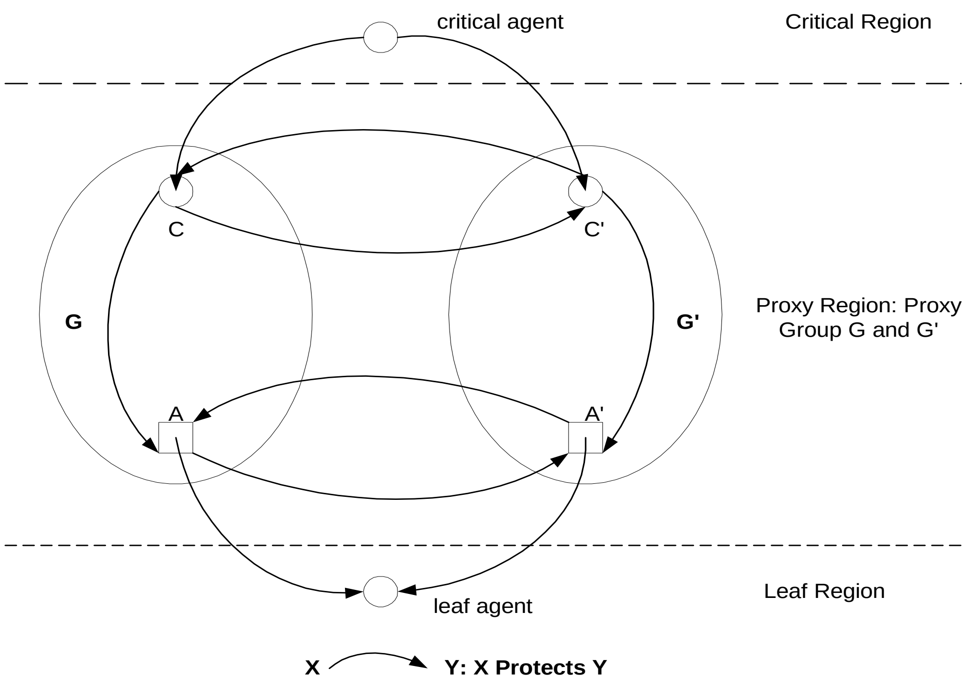

Figure 3-4: The protection scheme in the region

3.7 Procedures of location update and update/downlink

In this section, we will describe in detail the location update operation, the data flow (uplink) and command flow (downlink) procedures.

3.7.1 Location updates of leaf agents

Whenever a leaf agent changes its location, it must report its current location once to the multicast group G. Then the multicast group will receive these location update messages. The group coordinator C will forward these messages to its shadow C’ that will be responsible for the maintenance of location database. Lastly, C’ will distribute the location update information to the related agent in its group so that a part of the location database is stored in these agents.

3.7.2 Location updates of proxy group G and G’

The coordinator C for multicast group G will be responsible for the location update and join/leave operation in its group. In short, C will maintain a secure group G. The coordinator C’ for group G’ is the shadow of C in group G. C’ maintain the location update message of all its members. Note that G’ is not a multicast group so the communication inside the group is a one to one connection secured by conventional methods. C and C’ may store their current location for each other and also report their location to the critical server.

3.7.3 Location updates of critical agents

The critical agents will be managed by their critical server that not only stores their location update messages but also maintain the current location of C and C'. As mentioned before, the central directory server is replaced by several directory servers. They consist of critical agent server for the critical agents in the critical region, the proxy agents of group C' as the location server for all leaf agents in the leaf regions and C and C' for the two group of proxy agents.

3.8 Procedures of uplink

In this case, a leaf agent X wants to transfer some data to a critical agent Y. The procedure is shown in Figure 3-5.

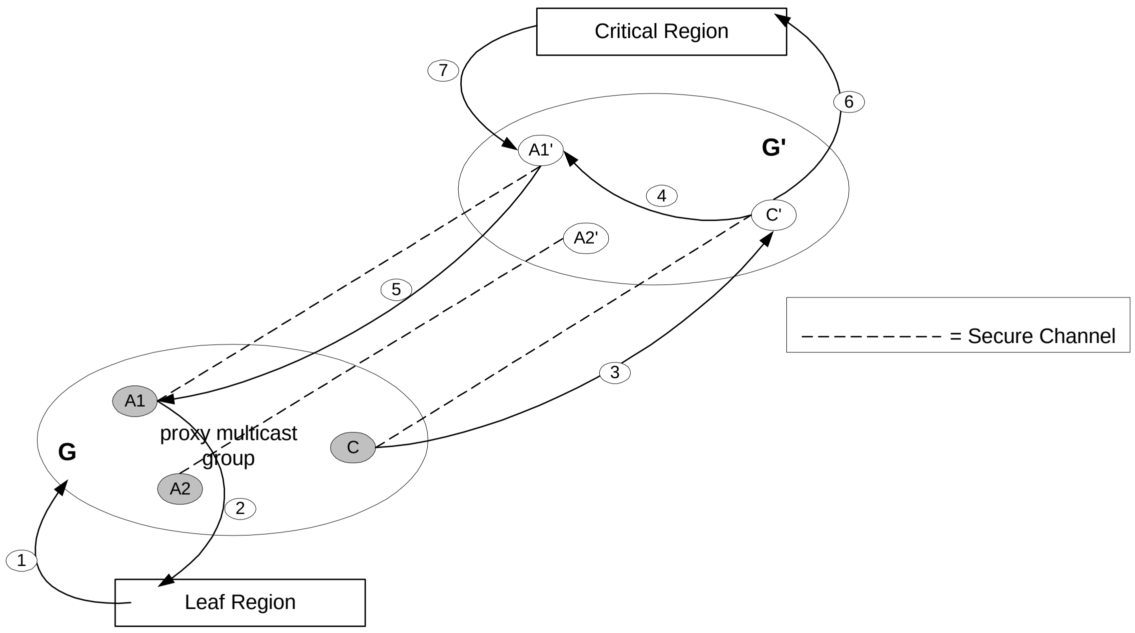

Figure 3-5: The data upload procedure

Step 1: A request for transferring data to critical agent is sent to the multicast group.

Step 2: The responding agent (suppose A1 responsible for this request) receive this request, it will initiate a connection to the requesting agent and get the data. All other agents in this group except group coordinator will simply drop this multicast message.

Step 3: The group coordinator C will initiate a connection to its shadow C' and inform of this data request. The identification of the responding agent (A1) is included in the message.

Step 4: The group coordinator C' then informs A1' (A1 "buddy") that a data request is coming.

Step 5: While A1' receiving this information will initiate a connection to A1 and get data originated from leaf region.

Step 6: C' will then inform the critical agent about the request and A1 location.

Step 7: The critical agent will connect A1 and get the data.

3.9 Procedures of downlink of command

In this case, a critical agent wants to give a command to a leaf agent Y. The procedure is shown in Figure 3-6.

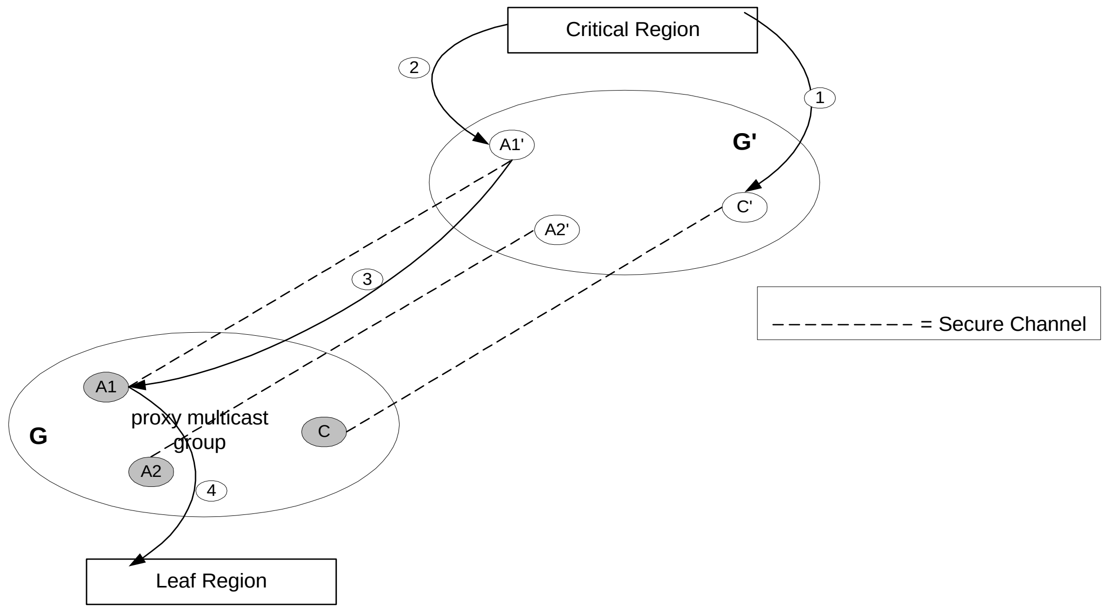

Figure 3-6: The commands download procedure

Step 1: The critical agent X first inform the coordinator C’ that it will send a command to a leaf agent and C’ selects a proxy agent A1 as the forwarding agent.

Step 2: The critical agent will then connect to the proxy agent A1’. A1’ will receive this request.

Step 3: A1’ initiate a connection to A1 (A1’ knows A1 current location). A1 get the command together with the leaf agent’s location.

Step 4: A1 initiates a connection to the leaf agent Y and transfers the command to it.

3.10 Attack Analysis on our model

We defined four kinds of attacks: the penetration attack (PA), the passive sniffing attack (SA), the active probing attacking (AP) and the denial of service attack (DoS). SA and AP are common attack techniques that gather information from local LAN before conducting PA or DoS attacks. Passive network sniffing is where an attacker listens to the network traffic passing by a host on which the attacker has control. Active probing is where a hacker maps out the hosts in a network by sending out packets to the IP addresses owned by an organization. Active probing can reveal hosts that are working, the operating systems they are running, the server applications running on these operating systems. The most popular software tool for active probing is Nmap.

3.10.1 Attack on leaf region (leaf agents)

The hosts in leaf region are penetrable. If a host is penetrated by the attacker, the attacker then use this host as the attacking point towards other hosts. For example, the attacker can use this host’s port to listen to the passing network packets and disable the whole LAN. Furthermore, the attackers can also use active probing to attack other hosts in the other LANs. If an attacker can penetrate many hosts undetected, we can clearly draw a conclusion that the IDS itself is not secure. In our approach, we proposed that the IDSs can still function and the attacker has not been successful in disrupting critical parts of the hosts.

An attacker can also destroy the IDS agent worked on this host. He or she may even modify the agent’s code for malicious purposes. In this case, the backup mechanism for the leaf agents can be used for resuming the ruined leaf agents. We can see from the last subsection that the attacker may get the location of a proxy agent during a connection. She or he will then launch an attack on the proxy region.

3.10.2 Attack on proxy region (proxy agents)

In our definition, we let the proxy agent change location after every operation with the leaf agent. If the attacker can still penetrate the host and he/she is fast enough, he/she still can control the host before the agent leaves the region. In this situation, the proxy agent in that multicast group will be in danger. The attacker will either disable the agents or wait in the host to sniff the network packets traffic. By doing this, the attacker will be able to determine the multicast group address and the proxy agent's buddy location. In the worst condition, the attacker breaking the proxy region may further attack the critical region that if successful, may cause some of the critical agents to malfunction, by resurrected by their backups. The agents move randomly in the region so that it cannot detect the exact location except for SA and PA. If an attacker launches a successful flooding DoS attack against a critical host, all critical agents will seamlessly move to another critical host.

In our assumption, the attacker will soon be detected before it launches another attack. The reason we can make such an assumption is because the hosts in proxy area are neither used as application clients nor as any important servers. Their tasks are simple enough so that it would be easy and straightforward to protect them. Also to detect the malicious status of these network elements may be very quick since they perform limited functions. We can at least protect the security of the proxy agent's "buddy". Attackers will then conduct a DoS attack if they cannot penetrate the critical hosts. Occasionally, the attacker randomly targets a host and hopefully it can spot a critical host.

Another method an attacker may use is traffic analysis. Since the attacker cannot sniff the other network segments, he or she can analyze the traffic of its penetrated segment. This segment may contain any one of these agents: the proxy agent, the buddy agent or none of the agents. For the proxy agent's segment, this traffic analysis cannot help the attacker to find out the group coordinator because all of the multicast messages are through secure group communication. For the buddy agent's segment, the attacker may find out the group coordinator since they communicate on a one to one scheme. If DoS attack is launched on the coordinator and cause it to shut down, its "buddy" agent may resume it by its knowledge.

3.10.3 Attack on critical region (critical agents)

With our proposed workable attack resistant properties, it is rather difficult for the attacker to penetrate the critical region. She or he also can not use sniffing or probing for some usable information since the critical agent does not reply on these random messages. The attacker may randomly select a critical host or get the IP address of some critical hosts by attacking the proxy agent and their buddies. He or she will also conduct DoS attack to them. The critical agents under such an attack will resurrect itself using its backup mechanism. This is to say that at least part of the IDSs system will not be completely down by such attack. If the IDSs can still function, system administrators will take action and disconnect the intruder. The IDSs will then be recovered by backup system.

3.11 Proposed Implementation Techniques

1. In our proposed solution, we have assumed that the agent system is actually an authenticated system. All agents arriving at the hosts are authenticated by their unique identifications and all packets arriving at the agents must be source authenticated. For example, the IDS hosts that an agent tries to visit will authenticate the agents' identities before they can host them successfully. A leaf agent may report its locations and send messages to a proxy agent. These agents who receive these packets must first authenticate their source.

The authentication scheme is used all over this architecture. It can be used to check if the agents stay in their proper regions (critical agent must stay in critical region, proxy agent must stay in proxy region and leaf agent must stay in leaf region). It can also be used to check if the packets are transferred to or from the right agents (e.g. there should be no connection initiated from a leaf agent to proxy agent or from proxy agent to a critical agent).

2. In our design, the proxy agents are formed inside a multicast group G. The main reason that we use multicast proxy group is to disseminate the central directory services to some distributed directory services. By this, we can ensure that the architecture has no central point of failure. With the backup scheme, we can quickly recover the ruined distributed directory services. The multicast group G is maintained by the group coordinator C. The responsibility of C is to ensure key refreshing at join/leave operations for securing the multicast group.

3. There are several steps for a traditional IDS architecture deploying our proposed scheme. Firstly, since our solution is actually mobile agent based architecture;

the traditional IDS system must first install a mobile agent platform for all IDS hosts. Secondly, the mobile agent system must be a secure one so that a mobile agent system must be secure one so that a mobile agent platform can be well protected. Thirdly, all the IDS components are wrapped into the proper agents. We have analysis agents (critical agents) put into the critical region and data collection or IDSs sensor agents (leaf agents) put into the leaf region. Next we must build our proxy agents and the buddy agents and form them into groups. Further on, all the agent activities are restricted by many security rules propose in our scheme.

As we did not change the IDS components from functioning unconventionally but instead we added in new features to enhance its robustness. We have wrapped them into agents and they are continued to detect intrusions and communicate with each other as usual. With all these assumptions proposed, we shall look into the provable attack properties.

3.12 Provable attack resistant properties in our model

This model has been designed to protect the critical IDS components of a traditional IDS hierarchy. In this section, we present proofs that describe the protections afforded critical IDS components. The main proof of this section is to claim that distributed IDS that follow the model are "attack resistant". Definition of attack resistant: - critical host will not be easily penetrated or have its location discovered by an attacker, secondly no critical agents will be easily disabled by an attacker unless the attacker can disable the entire backbone network in that domain.

Theorem 1: For an IDS that follows the model, attackers will find difficulty to penetrate any critical or proxy host.

Proof 1: The attacker can only attack the leaf region, the proxy region and the critical region. From the above analysis, in the bad situation, the attack to leaf region may cause some of the leaf IDSs agents to be down, but can be recovered soon by the proxy region. In the worse situation, if the attacker passes the leaf region, it can probably launch the attack to the proxy region that may cause some of the proxy agents to be down, but they can be resurrected by their "buddy" agents. The critical hosts communicate only with critical and proxy hosts. In the worst situation, the attacker breaking the proxy region may further attack the critical region and if successful, may cause some of the critical agents to malfunction, but they can be resurrected by their backups. If an attacker launches a successful flooding DoS attack against the critical host, all critical agents will seamlessly move to another critical host.

Theorem 2: For an IDS that follows the model, attackers cannot easily discover the location of critical host.

Proof 2: First we show that attackers cannot easily determine the location of a critical host by sniffing network traffic. All the critical and proxy hosts reside in backbone networks can only occur on the enterprise bus and the communication is encrypted. Therefore, it is unlikely for an attacker to discover the location of a critical host by using sniffing.

Next, we show that attackers cannot determine the location of a critical host by active network probing. Critical hosts are installed only in the backbone networks and attackers can only control hosts in regions or the enterprise bus. If the scanning host lies about its location and pretends to be a backbone host, then the scanning host cannot see the replies and thus cannot determine whether or not a backbone IDS host exists at the scanned location.

Chapter 4

Protecting Mobile Agent System

4.1 Background of Mobile Agent System Security

As our model employs mobile agents for protecting IDS components, these mobile agents may be the next target if attackers cannot effectively locate IDS critical components. Therefore, security is a fundamental concern for a mobile agent system. Harrison et. al[5] identify security as a "severe concern" and regard it as the primary obstacle to adopting mobile agent systems. The security problem of mobile agent system can be classified into two types: - protection of host resources, and protection of

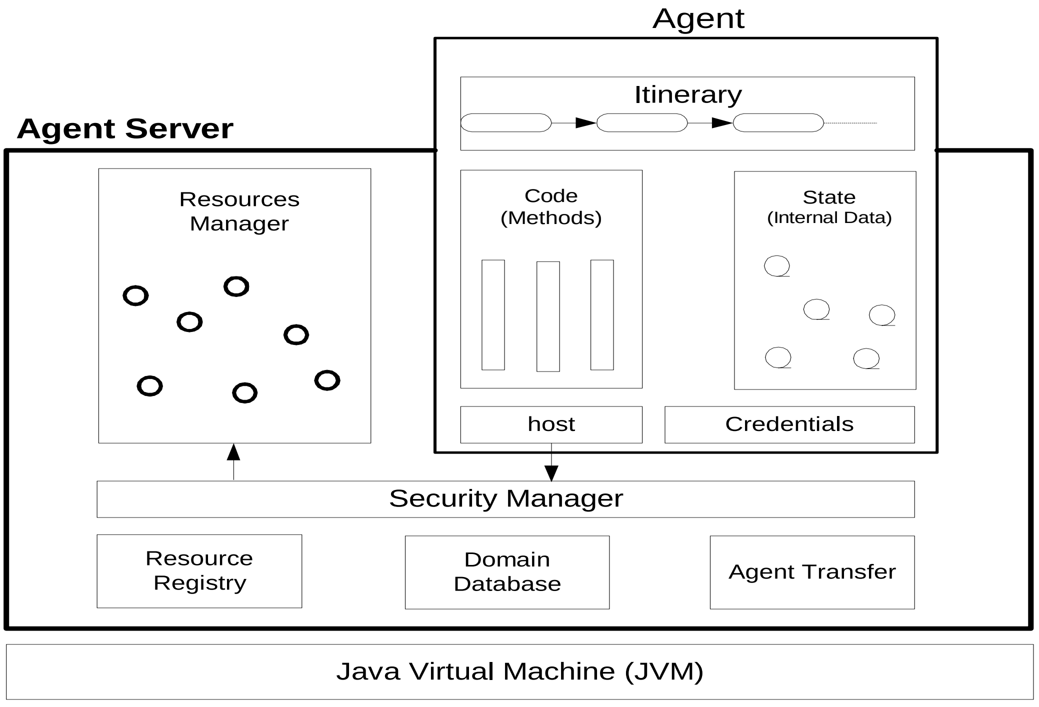

Figure 4-1: Schematics of Agent System using Java

agents. A host node will run one or more agent server processes (see Figure 4-1), which facilitate the execution of visiting agents. Each agent server has an agent environment component, which acts as the interface between visiting agents and server. The server's domain database keeps track of agents currently executing on it, and responds to status queries from their requestors. The agent transfer component implements a protocol that allows agents to migrate from server to server. A resource is an object that acts as an interface to some service or information available at the host. The server maintains a resource registry which is used in setting up safe bindings between resources and agents.

Recently, Java has emerged as a widely used basis for building mobile agent systems because of its support for object-orientation, its security model for mobile code, and a rich set of libraries which support object serialization. The Aglet Workbench[10] developed by IBM is an example of a Java-based mobile agent system. It uses an event-based programming model; handlers are defined for different types of events such as migration, dispatch, arrival at a server etc. The Java environment has a security aware design. Its security model has three main components: - a byte code verifier ensuring that programs do not violate type safety, a class loader objects are use to dynamically load classes into Java runtime, and security manager which can encode a security policy and perform some basic access control functions.

4.1.1 Protection of Host Resource

A host participating in a mobile system runs an agent server process. The host is exposed to various types of attacks, launched by malicious agents. These attacks can be categorized as: damage to host resources, denial of service to other agents etc. In order to prevent these attacks, we must protect the agent system by controlling the agent's access to system resource using security manager, wrapper[38] etc. At the same time, legitimate agents must be given access to the resources they need.

4.1.2 Protection of Agents

Many researches have been focused on some security issues of mobile agents, they have proposed framework of mobile agents called SOMA and Ajinta separately. These frameworks have similar functions and comprise of agent servers, management systems and security policies. In order to prevent malicious host's tampering. Farmer and Swarup proposed an authentication mechanism[43] to guarantee that the mobile agent only executes in a trust environment to prevent attacks. Some signature schemes[30] have been proposed to protect mobile agents' paradigm. In [41], two undetachable signature schemes are developed in RSA signature scheme and computed with encrypted functions. The undetachable signature scheme allows the mobile agent to sign a message without revealing the secret key.

An agent acts as a delegate of its creator application and executes on behalf of its owner. Each agent carries a set of credentials which compose the agent's identity and public key certificate. The creator may delegate to the agent only a limited set of privileges while working on its behalf. Such access restriction is encoded in the credentials. When a server receives an agent, it uses these credentials to validate the authenticity of the agent, and based on the agent's identity and delegated rights, it can grant access privileges for its local resources.

Based on Volker and Mehrdad’s scheme, they proposed a key management and access control mechanism. In their scheme, they constructed a mobile agent structure and designed a cryptographic key assignment mechanism to control the access of confidential data in a mobile agent. This provide a safe binding between the visiting agent code and the server resources, so that the agent can access the resources it needs but cannot breach system security by accessing the resources it is not authorized to use. However, their scheme is inefficient. It requires a larger agent size and higher computational cost. As we adopted mobile agent technology in IDS architecture, we need an efficient mechanism to protect the agent. Comparing with Volker and Mehrdad’s scheme, our scheme requires smaller amount of storage for the agent and lower computational cost for deriving the cryptographic keys.

4.2 A Review of Volker and Mehrdad’s Scheme

In 1998, Volker and Mehrdad constructed a structure for mobile agent and proposed a solution for access control and key management problems in a mobile agent by using cryptographic implementation approach. An efficient solution for controlling access in hostile environments is to encrypt the confidential data.

4.2.1 Volker and Mehrdad’s Mobile Agent Structure

The structure can support cryptographic encryption/decryption, agent access control, and key management for agents. Furthermore, the encryption/decryption mechanism and the digital signature mechanism can easily apply to the structure to ensure the confidentiality, integrity, and authenticity of the agent. Figure 4-2 illustrates mobile agent structure in a tree-based structure. The nodes of the structure might either be folders or files. The agent structure can be divided into two branches, the static branch and the dynamic branch. In static branch, the contents are all fixed data of the agent, such as class code, security policies. The features of these data are not changed during the life time of the agent. Whilst in the dynamic branch, the contents are variant data of the agent such as return results, access control keys. The branch contains a heap as a general storage space, and the security contents.

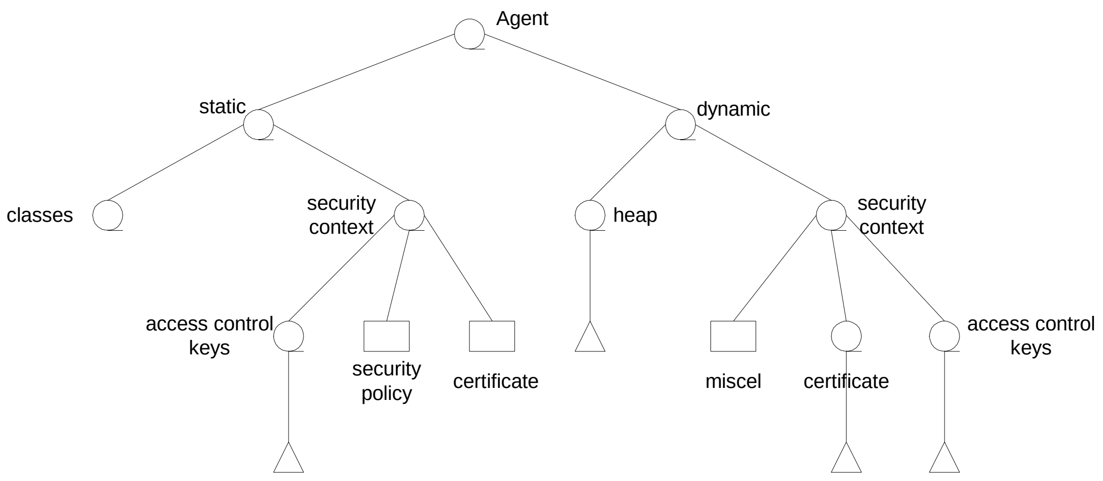

Figure 4-2: Volker and Mehrdad's Mobile Agent Structure

The agent structure can be efficient to achieve integrity protection and authentication by applying to a digital signature. The agent's owner can use digital signature to sign the static part. Thus the integrity of the context in the static branch can be verified by the hosts that the mobile agent is visiting. If the verification is correct, the visited host can trust that the agent's origination and its contents are not tampered. Additionally, when the agent has completed its tasks with a specific host, the return results are stored in the dynamic part. However, the previous collected data may be modified by the malicious hosts. For keeping the integrity, the malicious host has to sign the root node.

4.2.2 Volker and Mehrdad's Key Management and Access Control Strategy

In order to keep the confidentiality of the agent, Volker and Mehrdad has proposed a key management and access control strategy. The proposed strategy also can be done in the mobile agent structure efficiently. The objectives of their strategy are not only achieving the confidentiality of the security contents in both static and dynamic branches but also avoid malicious agents from accessing the secure contexts. Secondly, the access control keys are used to control the hosts' access to mobile agent. By doing so, we can prevent spying from malicious hosts. In the following, we shall briefly describe the key management and access control strategy.

First the agents use the symmetric cryptosystem to encrypt the all confidential files with enciphering keys $E_K = [E_{K1}, E_{K2}, \ldots, E_{KJ}]$. In symmetric encryption, the ciphertext is produced using the enciphering keys. Upon decryption, the ciphertext can be transformed back to the original plaintext by using a decryption algorithm and the same key that was used for encryption. Then the agent owner creates a history folder HF_i for each visited host VH_i within the static/security context/access control folder. Each folder HF_i contains the corresponding deciphering keys D_{Ki} according to the privileges of the visited host VH_i. If a host possesses the privilege to access a confidential file, it can find the deciphering key in its corresponding folder. In order to avoid the unauthorized user disclosing the confidential folders, each folder HF_i owned by the visited host VH_i is encrypted by an available public key cryptosystem with the VH_i public key E_{Ki}. Then, the encrypted results are placed into the *static/security context/*

access control folder. Therefore, when the agent arrives at a host, the host can find out its corresponding folder from *static/security context/access control* and each host VH_i only has the capability to access its corresponding folder by using its private key D_{ki}.

In Figure 4-3, a simple example of key management and access control strategy. In this figure, we assume that the folder of classes containing four files, which are agent.zip, negotiate.zip, support.zip, and updates.zip. The file agent.zip is non-confidential, but the other files are confidential and separately encrypted with the keys E_{k1}, E_{k2}, and Ek3. Furthermore, according to the privilege of the visited hosts, the agent owner creates three folders HF_1, HF_2, HF_3 which are on behalf of the proxy host, critical host, and leaf host separately. If they possess the privilege to access the specific files in the host, the corresponding deciphering keys are copied into the folder and encrypted by the host's public keys. For example, the folder HF_1 contains the encrypted keys E_{k1}, E_{k2}, E_{k3} and thus the proxy host can obtain the deciphering keys to access the files negotiate.zip, support.zip, and updates.zip. Similarly, the folder HF_2 contains E_{k2} and E_{k3} and thus the critical host can access the files support.zip and updates.zip. The folder HF_3 only contains K_3 and thus the leaf host can only access negotiate.zip. Therefore, the secret files can be protected by using the strategy of access control and key management.

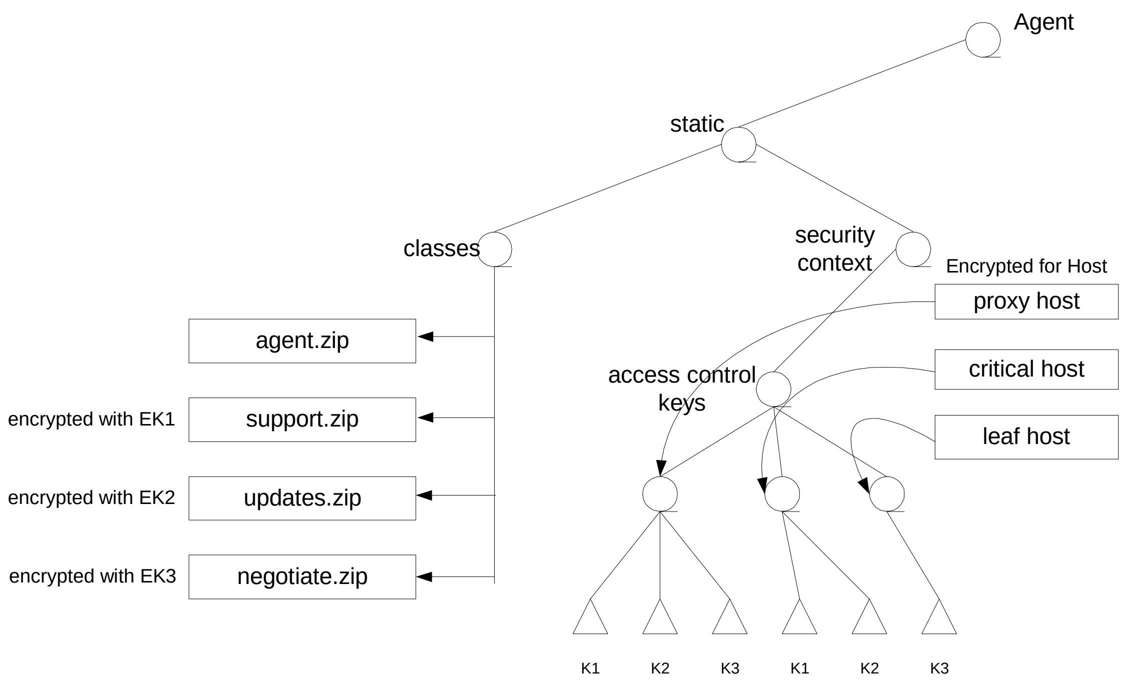

Figure 4-3: Volker and Mehrdad's access control and key management strategy

4.2.3 The Drawbacks of Volker and Mehrdad's Scheme

Volker and Mehrdad's scheme is an efficient tree-based structure with a key management and access control strategy for the mobile agent. In their strategy, each folder HF_i must hold and manage a set of subordinate keys. This arrangement raises the problems of large agent code and more public computations. It is better to make the size of the mobile agent small because a smaller mobile agent can conserve the network bandwidth. It is more easily delivered in the network. The followings are the drawbacks:-

1. Large agent code. In this scheme, we find that the decryption key is repeated to store in different folders. For example, K_1 can be found in both HF_1 and HF_3. The same thing goes for K_2 and K_3 too. It takes up too much space and increases the size of mobile agent.

2. More public key computation. Since decryption keys are repeated to store in the folder *static/security context/access control list*, the agent owner has to use more public key encryption computation to keep the folder secure, and the visited hosts also require more public key decryption computation to recover the decryption keys.

4.3 Proposed RSA Key Assignment and Access Control Strategy

Based on RSA public key cryptosystem[2], our proposed scheme plans to solve the cryptographic key assignment problem in mobile agent environments. In the section, we first present our strategy including key generation phase and key derivation phase. Then we present the dynamic key management problem in mobile agents.

4.3.1 The Proposed Scheme

The proposed strategy for performing key generation phase and key derivation phase are stated as follows:

Key Generation Phase

Step 1: The agent owner (system administrators) randomly chooses two large primes, p and q. For maximum security, choose p and q of equal length. Next, the agent owner calculates n such that n=p x q, where n is public.

Step 2: The agent owner chooses another parameter, g, which is relatively prime to n and in the range power of 2 less than n-1.

Step 3: The agent owner chooses a set of distinct primes $\{e_1,e_2.....,e_m\}$ for all confidential files $\{F_1,F_2,.......F_m\}$ where each e_i has to relatively prime to $\phi(n)$, i.e $\gcd(\phi(n), e_i) = 1$ and $1 < e_i < \phi(n)$, denotes the Euler's totient function of n.

Step 4: The agent owner calculates $\{d_1,d_2,.......d_m\}$, where each d_i is the multiplicative inverse of e_i, i.e $e_i \times d_i \equiv 1 \bmod \phi(n)$. The parameters $\{d_1,d_2,.......d_m\}$are kept secret, but the parameters $\{e_1,e_2.....e_m\}$ and n are published.

Step 5: The agent owner calculates the enciphering keys E_{k1}, E_{k2},..... E_{km} for encrypting the confidential files F_1, F_2,......F_m separately. The enciphering keys can be calculated by the following equation

$$E_k = g^{di} \bmod n \qquad (1)$$

Step 6: The agent owner generates the derivation keys $\{D_{k1}, D_{k2}.........D_{km}\}$ for all visited hosts $\{VH_1, VH_2,........VH_m\}$ as follows.

$$D_{km} = \ g^{\Pi}{}_{Fk<VH(dk)} \bmod n \qquad (2)$$

for k = 1,2......,m where $F_k < VH_i$ means that the visited host VH_i possesses a privilege to access the confidential file F_k. Next E_{ki} is encrypted by using the public key cryptosystem VH_i public key P_{Ki}. Therefore, the derivation key is kept secret. Only the host possesses the corresponding private key can discover the derivation key.

Key Derivation Phase

Step 1: When an agent arrives at a host VH_i, the host can find out its corresponding folder from *static/security context/access control list* and use its private key D_{ki} to decrypt the contents in its folder HF_i.

Step 2: If the host keeps the relation $F_m < VH_i$, the host VH_i can derive the deciphering key K_i from its derivation key D_{ki} as follows:

$$\begin{aligned} K_i &= DK_i^{\Pi_{Fk<VH,\, k \neq i}(e_k)} \bmod n \\ &= g^{\Pi_{Fk<VH(dk)}\, \Pi_{Fk<VH,\, k \neq i}(e_k)} \bmod n \\ &= g^{di} \bmod n \qquad (3) \end{aligned}$$

4.3.2 Implementation

In Figure 4-4, each host folder HF_i only contains a derivation key D_{ki} instead of many subordinate keys. According to the access policies, VH_1 possesses the greatest privileges. It can derive the three deciphering keys for encrypting the three confidential files. VH_2 possesses the privileges to access the files updates.zip and negotiate.zip. Thus it can derive the corresponding deciphering keys D_{k2} and D_{k3}. VH_3 possesses the least privilege. It can only access the file support.zip. Thus the derivation key is the deciphering key for decrypting the file support.zip.

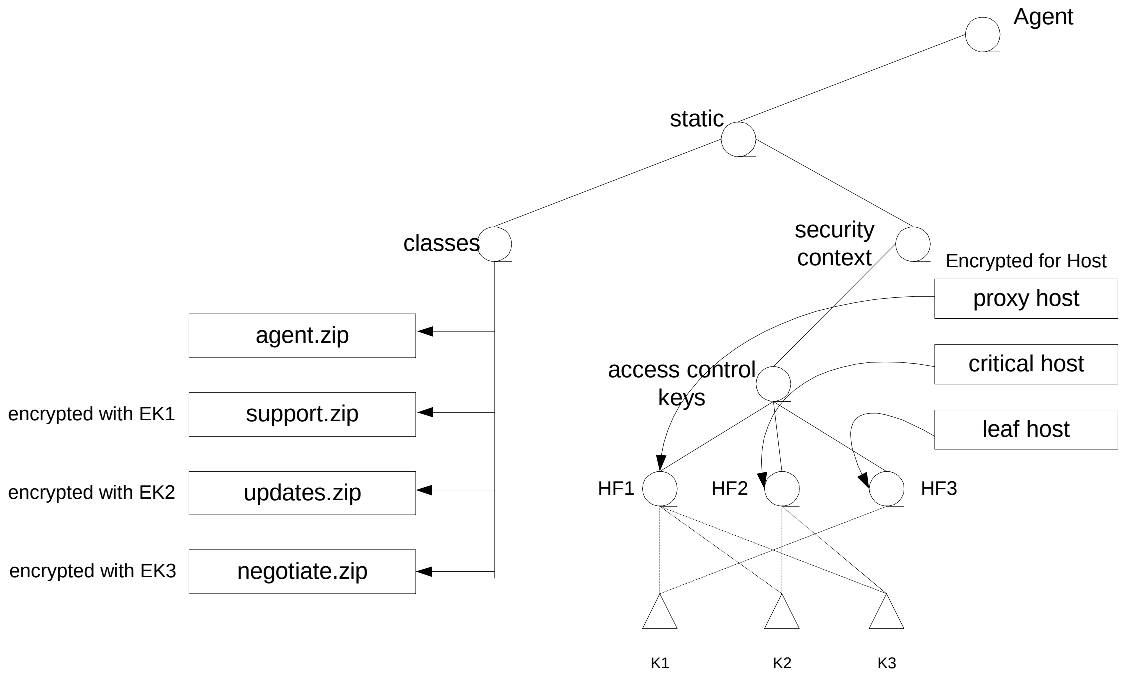

Figure 4-4: Our proposed strategy

In Figure 4-5, it shows the example of key assignment in our strategy. Assume that the agent owner first chooses two prime $p = 37$ and $q = 47$, and computes $n = p \times q = 1739$. Then, the agent owner chooses a parameter $g = 17$ which is relatively primes $e_1 = 5$, $e_2 = 11$, and $e_3 = 13$, and computes their multiplicative inverses $d_1 = 1325$, $d_2 = 1355$, and $d_3 = 637$. the enciphering keys E_{k1}, E_{k2}, and E_{k3} for the confidential files F_1, F_2, F_3 can be computed from Equation (1), and thus $K_1 = 338$, $K_2 = 1202$, and $K_3 = 392$. In addition, the agent owner also assigns the derivation keys to the visited hosts. Applying to the Equation (2), the derivation keys are computed as $D_{K1} = 17^{d1 \times d2 \times d3} \bmod 1739 = 1606$, $D_{K2} = 17^{d1 \times d2} \bmod 1739 = 949$ and $D_{K3} = 17^{d1} \bmod 1739 = 338$. Let the parameters e_1, e_2, e_3 and n public, and the other parameters p, q, E_{k1}, E_{k2}, $E_{k3,}$ $D_{k1,}$ D_{k2} and D_{k3} keep secret.

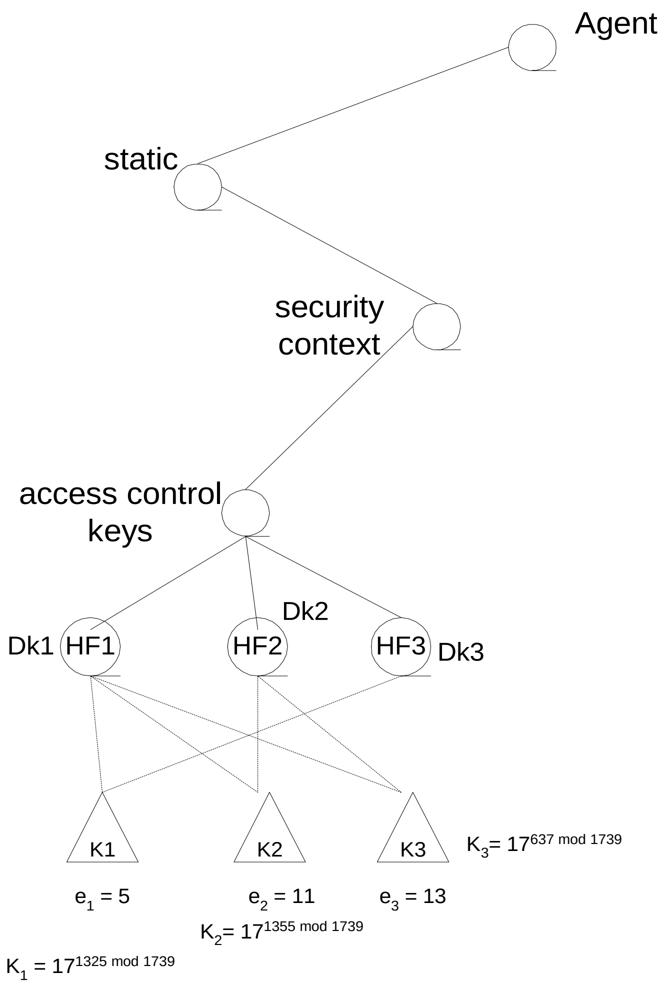

Figure 4-5: Key assignment in our proposed strategy

When the agent arrives at a host, the host can utilize its derivation key and the public parameters to calculate the deciphering key by means of the Equation (3). For example, VH_1 can obtain the deciphering keys K_1, K_2, and K_3, by calculating $K_1 = DK_1^{e2xe3} \bmod 1739 = 338$, $K_2 = DK_1^{e1xe3} \bmod 1739 = 1202$. Furthermore, if the host does not possess the privilege, it cannot gain the deciphering keys.

4.4 Dynamic Key Management

In many cases the agent owner has to adapt the agent dynamically. As a result, the system should be flexible enough to handle the dynamic key management problems. The problems are involved in the addition and deletion of confidential files, granting and removal of access right to a visited host. In the following, these issues will be considered.

4.4.1 Adding an access file

In an existing mobile agent structure, a new confidential file F_k is added to the agent. The agent owner has to perform the following works:

1. The agent owner establishes the access policies and chooses the parameter e_k and d_k for the confidential file F_k.
2. The agent owner updates the derivation keys D_{ki} for all the visited hosts VH_i with the access right to the new file. The updated derivation key D_{ki} can be calculated from the equation.

   $$D_{ki} = D_{ki}^{dk} \bmod n \qquad (4)$$

3. Store the secret parameter and publish the public parameter of F_k. Since the other public parameters and secret parameters are independent, they do not need to be changed.

4.4.2 Deleting an access file

When a confidential file F_k should be deleted from the agent, the agent owner has to perform the works:

1. Remove the confidential file from the agent.
2. The agent updates the derivation keys D_{ki} for all the visited hosts VH_i with the access right to the deleted file. The updated derivation key D_{ki} can be calculated from the equation

$$D_{ki} = D_{ki}^{ek} \bmod n \qquad (5)$$

Similar to the addition of an access file, the other parameters do not need to be changed.

4.4.3 Granting an access right

When the agent grants a new access relationship $F_k<VH_i$ to the VH_i, the agent will only updates VH_i derivation key DK_i. The updated derivation key DK_i can be calculated from the Equation (4). The other parameters also do not need to be changed.

4.4.4 Removing an access right

When the agent removes the access relationship $F_k<VH_i$ the agent requires to update VH_i derivation key D_{ki}. The updated derivation key D_{ki} can be calculated by Equation (5).

4.5 Security and Performance Analysis

We shall examine the security of our proposed key assignment and access control scheme in section 4.5.1 and 4.5.2. In addition, we shall also discuss the required storage and computational cost in our proposed scheme.

4.5.1 Cryptanalysis against RSA Security

Our proposed strategy is based on RSA cryptosystem. The security is similar to that of the RSA cryptosystem; it is based on the problem of factoring large number. Factoring n is the most obvious means of attack. However, without knowing the two large primes p and q, it is difficult to obtain the multiplicative inverse d_i from the public parameters e_i and the modular n. Any adversary who wants to derive the multiplicative inverse d_i has to first factorize n into its two prime factors.

It is certainly possible for a cryptanalyst to try every possible d_i until (s)/he stumbles on the correct one. Most discussion of the cryptanalysis of RSA have focused on the task of factoring n into its 2 prime factors. Determining ϕ (n) given n is equivalent to factoring n. With presently known algorithm, determining d given e_i and n appears to be at least as time consuming as the factoring problem.

For a large n with large prime factors, factoring is a hard problem, but not as hard as it used to be now. The threat to larger key sized is 2 fold with the continuing increase in computing power, and the continuing refinement of factoring algorithm. We can expect further refinement of generalized number field sieve (GNFS), and use of an even better algorithm is also possibility.

Thus, we need to be careful in choosing a key size for RSA. For the near future, a key size in the range of 1024 to 2048 bits seems reasonable. In addition to specify the size of n, a number of other constraints have been suggested by researchers. To avoid values of n that may be factored more easily. The algorithm researchers[26] suggest the following constraints on p and q:-

(1) p and q should differ in length by only a few digits. Thus, for a 1024 bits key (309 decimal digits); both p and q should be on the order of magnitude 10^{75} to 10^{100}.

(2) Both (p-1) and (q-1) should contain a large prime factor.

(3) gcd (p-1, q-1) should be small. In addition, it has demonstrated that if $e<n$ and $d<n^{1/4}$, and d can be easily determined.

4.5.2 Preventing unauthorized hosts from accessing

Without the privileges to access the confidential file F_i, the visited host VH_i cannot derive the deciphering keys K_i. Since the multiplicative inverse d_i is not embedded in the derivation key $D_{ki,}$ D_{ki} will reveal no information of F_i. Therefore, the malicious hosts have no way to derive the deciphering key from the derivation key and the public parameters. In the other hand, if an unauthorized host wants to derive the deciphering key $K_i = g^{di} \bmod n$ from the public parameters e_i and n, the problem is similar to factorizing the modular n.

4.5.3 Performance Analysis

In Volker and Mehrdad's scheme, it requires a larger agent size and a higher computational cost. In the scheme, the deciphering keys are stored repeatedly in the agent. Thus, a large agent size is required in their scheme especially when the agent carries large confidential files. This phenomenon raises a problem that the agent occupies more network bandwidth when it roams over the backbone. In order to reduce the overhead, we have to reduce the size of the mobile agent.

Assuming that a mobile agent will visit j hosts in the IDS network $VH_1, VH_2 \ldots\ldots VH_j$ and carry m confidential files $F_1, F_2 \ldots\ldots\ldots F_m$. Let the number of the files that the visited hosts are allowed to access are T_1, $T_2 \ldots\ldots\ldots T_r$. Consequently, let the length of the keys to range between 1 to n-1 in the used symmetric and public key cryptosystems. Therefore, Volker scheme requires to store $\sum_{i=1 \text{ to } r}^{Ti}$ deciphering keys in their corresponding folders and the total storage space of these deciphering keys is n x ($\sum_{i=1 \text{ to } r}^{Ti}$). However, only **r** derivation keys and m public parameters are required to store in our scheme. From the key generation phase, the public parameters and derivation keys are all between 1 to n-1. Thus only n x (r + m) memory space is required in our scheme.

On other aspect, how to reduce the computational cost is also an important issue. In Volker and Mehrdad's scheme, each host's folder in *static/security context/access control list* must be kept secret to avoid unauthorized host disclosing. Thus, each folder needs to be encrypted by using the public key cryptosystems. Since the plaintext length must be smaller or the same as the length of the public key, Volker and Mehrdad's scheme requires ($\sum_{i=1 \text{ to } r}^{Ti}$) exponential computation to decrypt the deciphering keys. In total, (2 x $\sum_{i=1 \text{ to } r}^{Ti}$) exponential computations are required in Volker and Mehrdad's scheme. However, our scheme only requires **r** exponential computations to encrypt the derivation keys, **r** exponential computations to decrypt the derivation keys and ($\sum_{i=1 \text{ to } r}^{Ti}$) exponential computations to derive the deciphering keys. Such that only 2**r** + ($\sum_{i=1 \text{ to } r}^{Ti}$) exponential computations are required in our scheme.

Chapter 5

Conclusions

5.1 Conclusions

Information has become critical to the success of business. Many businesses are built on information including software companies, brokerages, and advertising and marketing companies. Information may be more fragile than physical assets – it is more easily lost, destroyed, and stolen. With the emergence of distributed network environment, organization's critical information assets travel around the world on networks and in portable computers creating new challenges to the security of information.

In our model, it resists flooding DoS attacks using passive response system. Instead of trying to actively trying to stop an attacker's actions, our IDS model attempts to hide IDS components and move them away from harm. Thus our IDS components become invisible to an attacker's normal means of seeing in a network; passive sniffing, active network monitoring. The main functions of our proposed architecture are hiding critical IDSs agents and dissemination of central directory services. The success of these two functions will effectively protect critical IDS components. Despite the above mentioned, we make extensive use of mobile agent technology. This technology is crucial to the architecture because it provides backup capability for critical agents. Furthermore, this technology can allow critical components to randomly move around hosts.

Although many people avoid mobile agent technology because of its security trusts. This common perception has arisen because many people want to use mobile agents in E-commerce that do no necessarily trust each other. However, one can implement a secure mobile agent application by creating a "closed" mobile system. In a "closed" environment, all resource accesses would have to be checked by the security manager in the agent server. However, there are a few disadvantages. Because all security policies would have to be enforced through it, the security manager may tend to become an excessively large module and that could raise the potential for introducing errors during extensions. Therefore, our approach is to implement a public key cryptosystem to authenticate agents involved.

Despite these advantages and provable characteristics, our model is not completely secure. As it is impossible to build a completely secure system but at least it proves to be one effective method available for thwarting DoS attacks.

In conclusion, we envision IDSs of the future playing an increasingly prominent role in securing organizations' networks, both from the detection and response arena. Due to this, attackers will attempt to disable IDSs before penetrating and tapping more valuable resources.

5.2 Future Works

One difficulty for theoretical research in defending intrusion detection system has been the lack of proper models for the design. Most of the past research has been focused on the developments of intrusion detection system and how effective intrusion can be detected. In most existing intrusion detection work these decisions have been made implicitly by the designers.

In my research, I have started working on developing a model that will serve as a defensive system for intrusion detection system. This model represents how a defensive system are conceptually designed, implemented, deployed and used. There are two basic problems that we have to address in the future. The first is to design protocols for constructing credentials (enhanced Volker and Mehrdad's scheme) for an agent to act as its owners. Secondly, we will have to build a proxy mechanism in the proxy region. In this approach, this can be used to control binding between agents co-located at a proxy host, allowing them to securely communicate with agents in the leaf regions. And the proxy serving as a capability and its propagation can be restricted by encapsulating within it the identity of the agent to whom it was granted. We will also implement resource usage monitoring and selective revocation in the proxy region.

References

[1] B.Bobkiewicz. Hidden Backdoors, "*Trojan Horses and Rootkit Tools in a Windows Environment*", Jan 2003. http://www.windowsecurity.com/articles/Hidden_Backdoors_Trojan_Horses_and_Rootkit_Tools_in_a_Windows_Environment.html

[2] B. Jai Sundar, G.Fernandes, Isacoff, E.Zamboni. "*An architecture for intrusion detection using autonomous agents*", Computer Security Applications Conference, Proceedings 14th Annual, pages 13-24, 1998.

[3] B.Schneier. Applied Cryptography. New York, John Wiley & Sons, 2nd edition, 1996.

[4] C.A. Carver Jr, J. Humphries and J.M.D Hill. Real Time Intrusion Detection Systems. *Computer Networks 33*, 2000.

[5] CERT advisory CA-2000-01 on Tribal Flood Network and Stacheldraht Denial of Service Attacks.

[6] C.G.Harrison, D.M.Chess, and A.Kershenbaum. "*Mobile Agents: Are they a good idea*?" Technical Report, IBM Research Report, IBM Research Division, T.J.Watson Research Centre.

[7] C.Krugel, T.Toth. "*Applying Mobile Technology to Intrusion Detection*". *Technical University Vienna*, 2001.

[8] Cisco,NetRanger, http://www.cisco.com/univercd/cc/td/doc/product/iaabu/netranger

[9] C.Schuba, I.Krsul, M.Kuhn, E.Spafford, A.Sundaram, and D.Zamboni. "*Analysis of a denial of service attack on TCP". In Proceedings of the 1997 IEEE Symp. On Security and Privacy, pages 208-223,* May 1997.

[10] D.B.Lange and M.Oshima. "Programming and Deploying Java Mobile Agents with Aglets". *Addison-Wesley*. 1998.

[11] D.E.Denning. "*An Intrusion Detection Model*". *IEEE Transactions on Software Engineering, Vol 13(2), pages 222-232,* 1987.

[12] D.E.Denning, D.L.Edwards, R.Jagannathan, T.F.Lunt and P.G.Neumann. "*A prototype IDES – a real time intrusion detection expert system*", *Technical Report, Computer Science Laboratory, SRI International,* 1987.

[13] D.L.Pipkin. Information Security, Protecting the Global Enterprise. *Prentice Hall, Inc.* 2000.

[14] Fraser, B. (CERT). Private Communication, 1995.

[15] Forrester Online Retail Index. URL://http:www.forrester.com/NRF. Forrester Research Inc.

[16] G.Helmer, J.S.KWong, V.Honavar, and L.Miller. "*Lightweight Agents for Intrusion Detection*", Nov 2000.

[17] G.Helmer, S.K.Wong, V.Honavar and L.Miller. "*Intelligent Agents for Intrusion Detection*". *Proc. IEEE Information Tech. Conference, pages 121-124,* Sept 1998.

[18] J.P.Anderson Computer Security Threat Monitoring and Surveillance. Technical Report, 79F296400, J.P.Anderson Co. April 1980.

[19] J.Mirkovic, J.Martin, and P.Reiher. "*A Taxanomy of DDoS Attacks and DDoS Defense Mechanims*". Technical Report, 020018, University of California, LA, Sept 2000.

[20] K.A.Jackson, Intrusion Detection System (IDS) Product Survey, Los Alamos National Laboratory report, 6/25/99.

[21] L.Heberlein, G.Dias, K.Levitt, B.Mukherjee and D.Wolber. "*A Network Security Monitor*", *Proc. of IEEE Symposium on Research in Security and Privacy*, May1990.

[22] M.Asaka, S.Okazawa, A.Taguchi and S.Goto. "*A method of tracing intruders by use of mobile agents*", INET'99, June 1999.

[23] M.C.Bernardes, E.Moreira. "*Implementation of an Intrusion Detection System Based on Mobile Agents*", Proceedings International Symposium on Software Engineering for Parallel and Distributed System, pages 156-164, 2000.

[24] M.Crosbie and E.H.Spafford. "*Defending a Computer System using Autonomous Agents*". Technical Report CSD-TR-95-022, Coast TR-95-02. Department of Computer Science, Purdue University. 1995.

[25] M.Crosbie and E.H.Spafford. "*Active Defense of a Computer System using Autonomous Agents*". Technical Report CSD-TR-95-008. Department of Computer Science, Purdue University. 1995.

[26] M.Wiener. "*Cryptanalysis of Short RSA Secret Exponents*". IEEE transactions on Information Theory, vol IT-36, 1990.

[27] mixter.void.ru/tfn3k.txt, by Mixter

[28] P.E.Proctor. The Practical Intrusion Detection Handbook. *Prentice Hall, Inc.* 2001.

[29] R.Heady, G.Luger, A.Maccabe and M.Servilla. "*The Architecture of a Network Level Intrusion Detection System*". *Technical Report, University of New Mexico.* August 1990.

[30] P.Kotazanikolaou, M.Burmester, and V.Chrissikopoulous. "*Secure transactions with mobile agent in hostile environments*", *LNCS-Information Security and Privacy, Vol. 1841, Springer Verlag, pages 289-297,* 2000.

[31] P.Mell, D.Marks and M.McLarnon. "*A Denial of Service Resistant Intrusion Detection Architecture*". *Computer Networks(34), Special Issue on Intrusion Detection, pages 641-658,* 2000.

[32] P.Neumann and D.Parker. "*A Summary of Computer Misuse Techniques*". *In Proc.12th National Computer Security Conference, pages 396-407,* 1989.

[33] R.Bace. "*A New Look At Perpetrators of Computer Crime*". *In Proc. 16th Department of Energy Computer Security Group Conference,* 1994.

[34] R.Montanari, C.Stefanelli, and N.Dulay. "*Flexible security policies for mobile agent systems*", *Microprocessors and Microsystems, Vol 25, pages 93-99,* 2001.

[35] R.Papalus. Press Release: - Cyber Attacks Rise from Outside and Inside Corporations. April 1999.

[36] R.Volker and J.S.Mehradad. "*Access Control and Key Management for Mobile Agents*", *Comput. And Graphics, Vol 22, no 4, pages 457-467,* 1998.

[37] S.Kumar. Classification and Detection of computer intrusions, Ph.D Thesis, Purdue University, West Lafayette, IN47907, 1995.

[38] S.P.Liu, S.Y.Zhang, C.R.Wu. "*Wrapper based Security Monitoring Framework in Mobile Agent System*". *Journal of Software, Vol.12*, 2001.

[39] S.R.S et.al. "*DIDS – Motivation, architecture and early prototype*", *Proc. of 14th National Computer Security Conference,* Oct 1991.

[40] S.Savage, D.Wetherall, A. Karlin, and T.Andersson. "*Practical Network Support for IP Traceback*". *Proc. of the ACM SIGGOMM'2000, Computer Communication Review Vol. 30, No. 4, pages 295-306,* 2000.

[41] T.Sander and C.F Tschudin. "*Protecting mobile agents against malicious hosts*". *LNCS-Mobile Agent Security, Vol. 1419, Springer Verlag, pages 44-60,*1998.

[42] T.Y Li, W.M Chew and K.Y Lam. "*Defending against Distributed Denial of Service attacks using resistant Mobile Agent Architecture*", *Proc. of IEEE IPDPS 02, Workshop on Internet Computing and E-Commerce*, Apr 2002.

[43] W.Farmer, and V.Swarup. "*Security for mobile agents: Authentication and State appraisal*". *LNCS-Research in Computer Society, Vol. 1146, Springer Verlag, pages 118-130*,1996.

[44] W.Lee. "*A data mining framework for building intrusion detection models*". In IEEE Symposium on Security and Privacy, pages 120-132, Berkeley, May 1999.

[45] 1999 CSI/FBI Computer Crime and Security Survey. *Computer Security Issues & Trends*, Spring 1999.