International Journal of Computer Applications (0975 – 8887)
Volume 12– No.3, November 2010

An Agent based Distributed Security System for Intrusion Detection in Computer Networks

Arun Saxena
Amity University
Malhor Campus
Lucknow, India

A.K.Sharma
Y.M.C.A. Institute of Engineering
Sector-6
Faridabad, India

ABSTRACT

Intruders damage or steal valuable information by either bypassing security tools or penetrating through them, necessitating the need to detect such intrusion attempts especially in case of multi-event based distributed attacks spanning over a considerable amount of time. In this paper ecology of agents is being suggested that uses class hierarchy to define complex intrusions. The source and target containers produced thereof are analyzed for possible intrusion attempts thereby rendering the proposed system self-monitoring, robust, secure and reliable.

Keywords

Agents, Agent Based Intrusion Detection System, Distributed Intrusion Detection System, Network Security.

1. INTRODUCTION

Even after adapting best protection practices network security remains a challenge for system administrators because unknown security bugs always exist in a system, network configuration is continuously changing leading to creation of unknown security bugs and the easy availability of sophisticated tools and techniques to attackers for exploiting a system. To counter these challenges continuous monitoring and supervision of the infrastructure and security tools used is required.

Intrusion Detection Systems (IDS) detect and possibly prevent activities that may compromise system security, or an intrusion attempt in progress including reconnaissance/data collection phases that involve for example, port scans. The unusual activity detected is notified to the administrators so that a suspected connection can be blocked [1].

Intrusion detection can be broadly classified into two categories [30]. First is misuse detection model where detection is performed by looking for the exploitation of known weak points in the system, which can be described by a specific pattern or sequence of events or data. Second is anomaly detection model where detection is performed by detecting changes in patterns of utilization or behavior of the system. It is performed by building a model that contains metrics derived from system operation and flagging as intrusive any observations that have a significant deviation from the model [31].

The above models can be applied to either host-based systems or network-based systems. Host based systems base their decisions on information obtained from a single host (usually audit trials), while network based systems obtain data by monitoring the traffic of information in the network to which hosts are connected [30]. Further IDS can be arranged as either centralized (physically integrated with a firewall) or distributed. A distributed IDS consists of multiple IDS over a large network, all of which communicate with each other. More sophisticated intrusion detection systems are created using agent structure principle or multi-agent architecture.

Use of agents and mobile agents can improve the performance of IDS in several ways [20][24]. In this paper a distributed agent based architecture following an object-oriented approach to detect intrusions is proposed where a collection of agents is used to determine any ongoing intrusion attempt. Each agent has an independent job to do and is monitored by a designated agent so that any attempt to compromise the system itself can be quickly identified, a deviation from other existing systems as it does not use a hierarchy of agents to identify intrusions.

The rest of the paper is organized as follows. Section 2 covers the related work done in this area. Section 3 covers comprehensive explanation of the proposed system. Section 4 explains the capabilities and working of the system with the help of examples followed by conclusion.

2. RELATED WORKS

Intrusion detection can be traced back to publication of a technical report in 1980 [32] and has become a well-established research area after the introduction of a model [31]. These IDS are available as research prototypes or commercial products. Early example of these IDS includes ASAX [29], DIDS [22], NSTAT [26] and Net STAT [21]. The drawback of these systems is their limited scalability due to their centralized nature.

To overcome scalability limitation of these systems, later systems such as Emerald [25], GriDS [27] and AAFID [24] deployed IDS at different locations and organized them into a hierarchy such that low-level IDS’s send designated information to higher level IDS’s. The main problem with such an approach is that for two or more IDS’s which are far apart in the hierarchy detect a common intrusion, the detection cannot be correlated until messages coming from different IDS's reach a common high level IDS. This will require messages to traverse multiple IDS’s resulting in communication overheads.

The CIDF [23] goes one step further as it aims to enable different intrusion detection and response components to interoperate and share information and resources in distributed manner. CARDS [19] another IDS aims at detecting attacks that cannot be detected using data collected from any single location CARDS decompose global representation of distributed attacks into smaller units that corresponding distributed events indicating the attacks. It then executes and coordinates the decomposed smaller units in places where corresponding units are observed. Also in CARDS one component sends message to another only when the message is required by the later IDS to detect certain attacks. The communication cost is therefore reduced. DSOC [7] is another security architecture in which a local intrusion detection engine analyzes the data collected by one or several data collection boxes to find the intrusion patterns. Afterwards a global intrusion detection engine to find more complex intrusions and to give a global view of network security processes all generated alerts. However the functionalities of the global analyzer are not defined and security of individual elements is also not addressed exhaustively.

In the recent years many intrusion detection systems making use of agents and mobile agents have been proposed. Laocoonte [3], a novel mobile based distributed intrusion detection system [5], LAMAIDS [6], mobile agents for network intrusion resistance [8], design of a multi-agent based intelligent intrusion detection system [9], IDSUDA [12], MSABIDS [13], agent based network intrusion detection system [14], APHIDS [15], an adaptive intrusion detection and defense system based on mobile agents [16] and intelligent and mobile agent for intrusion detection system [18] are examples of systems which make use of mobile agents to perform distributed intrusion detection. However there are many disadvantages [20] of using agents and the mobile agent based systems have failed to address them.

A critical look at the available literature [20] [17] [11] and an early report [28] laying the requirements of an IDS indicate the following issues related to an IDS that needs to be addressed:

- Agents used in an IDS must be protected against malicious code, modification or eavesdropping during their transit over network.
- Code size of the agents must be small so that they incur less communication overheads and can be transferred in small amount of time over the network.
- Agent code execution must be fast.
- It must have a generic structure.
- It must be efficient in the way that it obtains audit data from various distributed sources and distributes the information processing and intrusion detection effort.
- It must be upgradeable, maintainable and easy to test.
- It must meet performance benchmarks.
- It must run continually without human intervention.
- It must be fault tolerant in the sense that it must be able to recover from system crashes and initialization.
- It must resist subversion. The IDS must be able to monitor itself and detect if an attacker has modified it.
- It must impose a minimal overhead on the system where it is running.
- It must be able to be configured according to the security policies of the system that is being monitored.
- It must be able to adapt to changes in system and user behavior over time.

All these issues except the last have been addressed and the proposed work is given in the next section. The proposed system also provides an automated approach to handle security events that are irrelevant to the targeted host by efficiently deducing false positives using system profiling [2].

3. PROPOSED SYSTEM

The proposed system uses two components namely, LIDM (local intrusion detection module) and GIDA (global intrusion detection agent). LIDM is deployed at a local site of the network if it has multiple segments and is responsible for detecting intrusion, raising alerts at the site where it is installed. It also analyzes the data of one of its neighboring sites. GIDA collects alerts from LIDM of various sites and identifies distributed attacks attempted from various sites. It also receives a message from any LIDM if an attempt to compromise it has been done and facilitates the recovery of the crashed LIDM. However it is not necessary to install both components if the network is not divided into subnets or segments. In such case only LIDM may be installed to achieve the desired objective.

3.1 Local Intrusion Detection Module

LIDM contains various agents to do specific tasks, namely

- Data Collection Agents (DCA)
- Master Data Collection Agent (MDCA)
- Remote Data Collection Agents (RDCA)
- Analyzing Agent (AA)
- Critical Data Analyzing Agent (CDAA)

Data from various sensors of the site is collected by DCAs and send to AA of that site for analysis after converting it into a standard format. MDCA is responsible for spawning various DCAs and itself is spawned by AA of the site. RDCA is responsible for collecting data from critical sensors only and gives this data to AA of a neighboring site for analysis. CDAA is responsible for analyzing the data of a neighboring site. The alerts generated by CDAA are sent to AA and alerts generated by AA are sent to GIDA for storage and further analysis. The composition of LIDM is show in Fig 1 and detailed description of these agents is given subsequent sections.

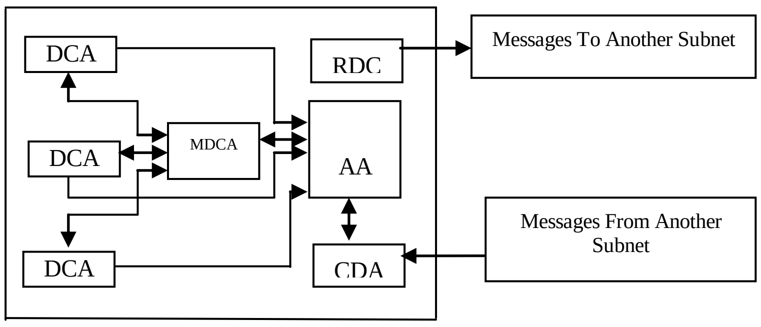

Fig.1 The Component Agents Of LIDM

3.1.1 Data Collection Agent (DCA)

The implementation of this system requires it to capture various events generated by sensors like firewall. This requires creation of several classes corresponding to the sensor events that may be generated. The event classes should provide for methods so that event listeners can be associated to them. DCA is a listener that is associated with several event types.

The functions of this component include sensing events, identify the protocol to which the event belongs, converting them into message object and sending it to AA. It is a simple reflex agent whose agent function is implemented using a table to store the percept sequence (i.e. event type) and routine names corresponding to source type to perform correct conversion. Agent performs a table lookup to call the correct routine for converting the message into standard format according to protocol. This approach is feasible because the size of the table required for this purpose is not too large. This agent would also require a parser, which would parse the incoming message to extract information from it. The extracted information would be used in the creation of message objects.

The log information gathered by DCA may be large and obtained at a high speed so to synchronize DCA with the log generation rate of sensors use of a buffer is recommended.

Whenever a new DCA is started by MDCA it is given a table containing various event names, protocols and their corresponding routine names along with the list of network addresses that it has to monitor. This list of addresses is stored in a configuration file by DCA and is used by DCA to receive only the messages, which are meant for it.

Any DCA must be able to monitor itself and report to MDCA if it is modified or an attempt to modify it has been done. So if attacker tries to read or write the configuration file, DCA sends a message containing information about the incidence to MDCA, which takes appropriate action [described in 3.2] depending on agent's current configuration. To further strengthen the security of DCAs MDCA polls all DCAs at short intervals and checks the contents of configuration file and agent function table, this would facilitate detection of attack on any DCA if for some reason DCA is not able to communicate MDCA that it is tampered.

Standard Data Objects Used In System

It can be easily observed that data would be collected from different sources and transmitted via different protocols. Hence the system requires standard objects to give source and target information, sensor information, protocol details, port details and standard message object to provide information about intruders to different components of the system. The class diagrams of these standard objects and their relationships have been discussed in this section.

Source And Target Objects

For intrusion detection purposes the source and target objects should be uniquely identified. The need for standardization of these objects appears because sensors may transmit host information in the IP address format or FQDN (Fully Qualified Domain Name) format, multi homing techniques provides multiple addresses for the same physical system, virtual host techniques provides multiple FQDN for the same physical system and finally reverse DNS lookup may be performed for each new (IP address) FQDN detected in logs [7]. Besides unique identification protocol and port information have to be added to these objects, as it would be useful in analysis of intrusions later by intrusion detection engine. These objects have same attributes except for their type so a single host class would be sufficient to define them. This class is a subclass of component class (explained in section 3.1.4.3), which has a tag attribute. The value of this tag attribute id used to distinguish between the two objects. The class diagram corresponding to standard host object is shown in Fig 2.

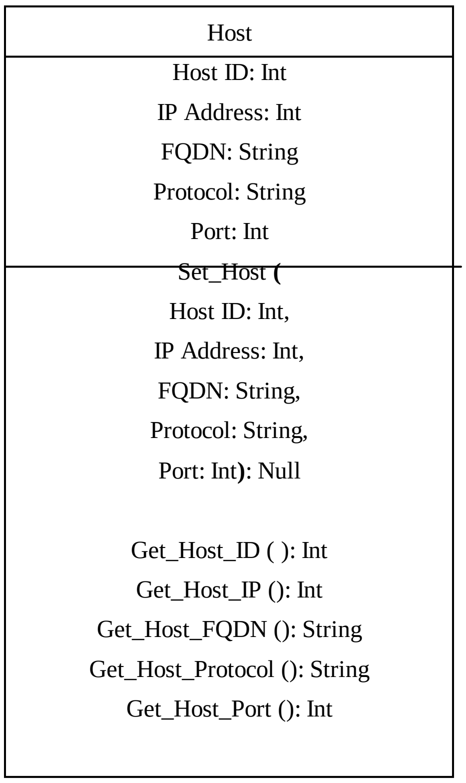

Fig. 2 Class Diagram Of Host

Message Object

The message object contains reference to sensor object, source object, target object and user object. Besides these the message object has attributes for unique message identification, date, time, a description of intrusion attempt as given by logs and intrusion type. The sensor_id attribute of sensor is a unique number given to each sensor for identification purpose. Other attributes of sensor object include sensor name and sensor description. The source and target objects can be created from

the same host class as explained earlier. User object gives information about the user who has performed the action and has two attributes namely, user name and real name.. Other attributes of message object are message id, message type, date, and time info and intrusion type. The function and implementation of objects, their attributes and methods is trivial and should not need further explanation. The class diagram in Fig 3 shows relationship between this object and other objects.

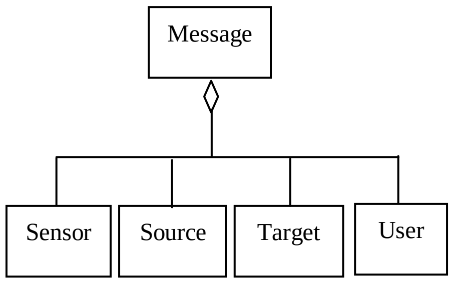

Fig. 3 Composition Of Message

3.1.2 Master Data Collection Agent (MDCA)

The function of this would be to oversee the operations of all DCAs, exert control over running DCAs by having the ability to start agents, stop agents and send configuration to agents. The configuration information can be stored in a configuration file.

At the system startup AA generates a configuration file containing addresses of the network to be monitored and gives it to MDCA. On the basis of this file the MDCA determines number of DCAs that would be required to monitor the network. This is advantageous as it deploys the DCAS according to the network size to be monitored and the number of messages that a single DCA can handle. It then assigns a set of addresses to each DCA for monitoring.

MDCA has to ensure that all DCAs work according to the configuration set by it. So upon receiving message about a read or write attempt from any DCA on its configuration file, MDCA should stop that DCA, reconfigure it and restart the monitoring.

AA starts a MDCA using a routine to create it first and then to configure it according to networks requirement. If the attacker tries to read this routine or modify it or perform a read or write the configuration file, MDCA sends a message to AA, which after checking the current status of MDCA reconfigures it, if required. AA also polls MDCA at regular intervals to detect any changes made MDCAs configuration by matching its current configuration against its original configuration.

3.1.3 Critical Data Collection Agent (CDCA)

This agent is same as DCA but is different from the DCA in the sense that it collects data only from critical sensors and from sensors hosting security tools on any site. This sends the received message to the CDAA of another site, which are used to give approximate security level of the concerned site in real time.

3.1.4 Analyzing Agent (AA)

This is the knowledge-base agent, which is responsible for intrusion detection on any site by analyzing a sequence of message objects. It receives these message objects from several DCAs. The message objects can be stored in a buffer from where the latest received object goes to the duplicates identification engine. The duplicates identification engine checks whether this new object is unique or not. If the object is duplicate then it discards it otherwise it sends it to intrusion detection engine for further processing and stores this object in message object repository. The intrusion detection engine analyzes the message object and creates objects used in intrusion detection and intrusion detection process is described in subsequently in this section.

Another job of AA is to compact alerts by merging similar ones. When attacked by intruder it sends a signal to GIDA, which creates a replica of this AA. Other functions that GIDA will perform when an AA is attacked are explained in section 3.2. AA consists of a local knowledge base, intrusion detection engine and an analysis engine. The architecture of AA is shown in Fig 5.

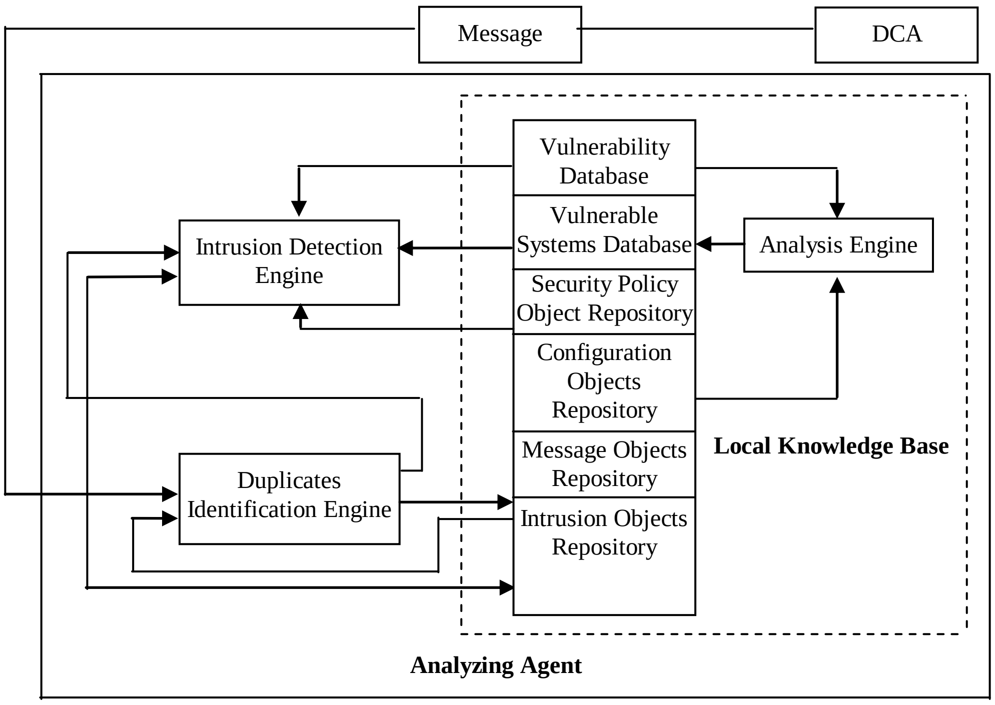
Message
DCA
Vulnerability Database
Vulnerable Systems Database
Security Policy Object Repository
Configuration Objects Repository
Message Objects Repository
Intrusion Objects Repository
Intrusion Detection Engine
Analysis Engine
Duplicates Identification Engine
Local Knowledge Base
Analyzing Agent

Fig. 5 Structure of Analyzing Agent

3.1.4.1 Local Knowledge Base

It contains information about known security breaches systems, which are exposed to various vulnerabilities, configuration of systems and security policy related information. A detailed discussion of these contents is as follows

Vulnerability Database

This database holds objects, which contain information about security breaches and insecure behavior that would either impact the overall security, level or that could be exploited by an attacker in order to perform an intrusion. The database format must make it possible to include structural vulnerabilities, functional vulnerabilities and topology-based vulnerabilities [7]. Moreover the database format must accommodate newly discovered vulnerabilities. Moreover this should be consistent with class hierarchy discussed in section 3.1.4.3. This database can be created by analyzing intrusions available from various sources like CERT, Bugtraq, and ArachNIDS etc.

Configuration Objects Repository

This object repository contains various resources of organization and their configuration details. Various objects in the database should represent resources/resource catalogs, their capabilities and some qualitative or quantitative value signifying the importance of these assets. The database format must make it possible for addition or modification of objects representing resources of the organization. However vulnerability systems database has to be updated to reflect changes in the configuration of the system.

Vulnerability Systems Database

This database contains a detail of exposure of various resources or assets to different vulnerabilities and the impact that that an intrusion can have on that resource and other resources attached to it. The creation of this database is done by analysis engine by correlating objects in vulnerability database and configuration objects repository.

Security Policy Objects Repository

While the above three components of local knowledge base represent the technical inventory of the organizational assets, this repository represents the structural inventory of the organization with the privileges given to users at each level. This helps in distinguishing between the legitimate users such as system administrators and an unauthorized user who tries to access the system to misuse it or steal important information from it.

Message Objects Repository

It is a collection of messages whose structure is already explained. Some criteria for deleting the messages in this repository can be defined on the basis of network traffic, which is being monitored.

Intrusion Objects Repository

This is a collection of objects created during ongoing intrusion detection process and objects, which signify an intrusion attempt. It not only stores the intrusion objects of its own subnet but also contains intrusion objects corresponding to intrusion attempts at the other subnet, which is being analyzed by CDAA of this subnet.

3.1.4.2 Analysis Engine

Its purpose is to correlate vulnerability database and configuration objects repository to produce vulnerability systems database. This vulnerability systems database produced contains details of configuration of different systems in the organization and their exposure to various existing vulnerabilities. This database helps intrusion detection engine in determining whether to create a container object corresponding to the newly received message object. It will create an object only if the source system is vulnerable to intrusion attempt indicated in the message object. This may help in reduction of false alarms generated by the system.

3.1.4.3Intrusion Detection Engine

Its purpose is to analyze messages to determine intrusion attempts. It has to create intrusion objects, create role object, add role objects to intrusion objects, add intrusion objects to appropriate container object, perform port and protocol analysis, perform time and date pattern analysis, perform system exposure and analyze the impact of intrusion and perform security policy matching. It also has to decide about the status of an intrusion object.

Various features of language to define an intrusion suggested in Common Intrusion Specification Language (CISL) [23] could also be met using an object-oriented approach. The suggested object oriented approach requires a class hierarchy. One or more objects participate in an intrusion attempt and each of these objects performs a specific role in the same. These would be referred to as role objects hereafter. An abstract class at the top of the class hierarchy has to be defined which encapsulates all general attributes of the role objects. This class is referred as component class in the suggested class hierarchy. All types of role objects that may exist should be then defined as subclasses of the component class. Any new role object detected can be easily added to the existing class hierarchy as a subclass of component class. This allows for accommodation of new role objects corresponding to newly discovered intrusions. Since the topmost class in the hierarchy is abstract so all its subclasses would necessarily inherit the mandatory attributes of an intrusion. One attribute of component class is tag attribute, which gives semantic clue about the role object. Since the abstract class at the top of the hierarchy has this attribute so all its subclasses will necessarily have it. Host object, sensor object and user objects are subclasses of this class.

The next class in the level of class hierarchy is referred as container hereafter. It is a subclass of component class and provides additional methods that allow an object of this class to contain various role objects. Moreover several container objects can be stored in a container object since they themselves are instances of container and this allows for multilevel containment system.

Next in the class hierarchy are four subclasses of container class. These are basically four different types of containers meant for different purposes namely intrusion, source container and target container. The tag attribute of an object of intrusion class gives a semantic clue about the intrusion attempt and other role objects i.e. contained within it will give details e.g. source, time, user etc of this intrusion attempt. Source container object will contain all intrusion objects matching a source and target container object will contain all intrusion objects matching a target. The class hierarchy is analogous to the one used by Java AWT and is shown in Fig 6.

Intrusion Detection

It is carried out in three phases: -

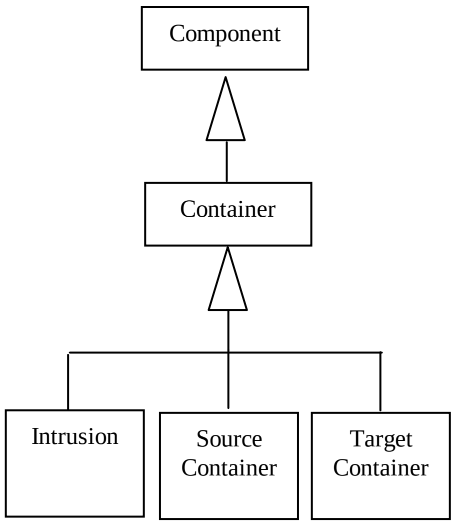

Fig. 6 Class Hierarchy For Container Objects

Phase 1: Object Creation, Protocol, Port and Time Analysis

This first step in intrusion detection involves the creation of an intrusion object on the basis of intrusion_type field of the message object. The tag field of intrusion object will contain the value of intrusion_type field of the message object. Then one or more role objects are created and added to this object of intrusion class. The tag attribute of the role objects defines the role of the object and other attributes describe the object itself.

After the creation of intrusion object the source object contained within this object is compared with the objects in the queue of active source objects if a match is found then this source object is added to the source container object corresponding to the matching source object otherwise target object of this intrusion object is compared with objects in the queue of active target container object so that it can be added to the matching object. If the newly created intrusion object does not match with any of the objects in source container object and target container object queues then a new source container object can be created for further analysis.

This newly created object or the source container object or the target container object should be subjected to protocol and port analysis. This analysis will help to identify various steps of intrusion e.g. a port scan followed finger printing /version identification on open ports followed by an exploit launched on vulnerable systems.

After protocol and port analysis these objects are subjected to an analysis based on time. This is used for slow and distributed intrusion processes e.g. a distributed denial of service attack against a same host and occurring at the same time can be determined using a time based analysis.

An important aspect of source container or target container object created is its status. An intrusion object can be active, inactive or closed. An active object is one, which matches the criteria defined for ongoing intrusion attempt. An inactive intrusion object is one which either does not match the criteria defined for active object or has not received a specific closure code. This means that this object is not being currently analyzed but can be reactivated by the next message matching the same object criteria. Objects having inactive status can be used to determine slow and distributed attacks. A closed object status is one whose processing is completed. Whenever a new container object is created upon receiving a message its status is set to be active.

When an object goes into closed state it is used to generate alert objects. These alert objects are send to global analyzing agent for permanent storage and to user interface for user's knowledge or for dissemination. Lastly the intrusion object is stored in intrusion objects repository.

Phase 2: System Exposure Analysis

This step is performed on the basis of information provided in phase 1. It evaluates the system exposure to the identified vulnerability and overall impact of such an intrusion attempt on the system. This is particularly required so that our system does not raise false alarms in case the vulnerability does not affect any resource. In case of unknown intrusion objects functional analysis facilitates addition of newly discovered vulnerability-to-vulnerability database if it can affect the system(s) under scan.

Phase 3: Security Policy Analysis

This is to differentiate legitimate users and preprogrammed audits, port scans etc. from intrusion attempts. This is similar to structural analysis in the sense that it is also sequence pattern matching which is performed by creating an object from intrusion object carrying relevant information to be matched against security policy objects stored in security object repository.

3.1.5 Critical Data Analyzing Agent (CDAA)

This is a child agent spawned by AA at each site to analyze messages send vide CDCA of another site. It has all the capabilities of AA and sends the generated intrusion objects to its parent AA which stores this intrusion object in its intrusion object repository and can use this information to detect distributed attacks at the concerned site. CDAA reduces the load of AA of remote site, which is required, as AA has to analyze complex intrusions at its own site.

In case an attacker tempers this CDAA it sends a message related to this to its parent AA that after checking the current status of CDAA creates a new agent and installs it, if required.

3.2 Global Intrusion Detection Agent

The structure of GIDA is shown in Fig 7. It receives various alerts generated by different LIDM and stores them in its knowledge base through alert knowledge base interface. It analyzes alerts stored in its knowledge base to detect intrusions occurring due to coordinated attacks and stores the generated alert on distributed alert repository. It also

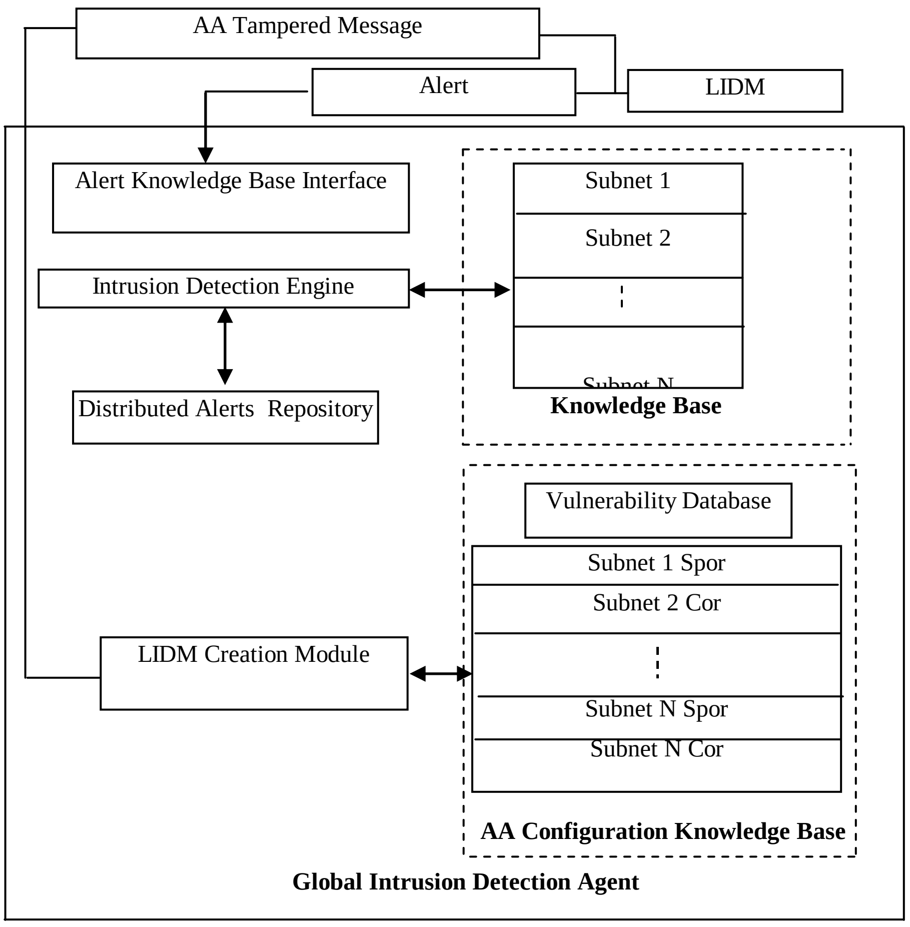

Fig. 7 Components of GIDA

merges them if possible to generate optimized outputs. It assigns a unique number to each subnet where an LIDM would be employed and LIDM are required to specify the subnet number in the alert sent by it. This arrangement is required to store alerts in an organized way so that they can be searched at a faster rate when coordinated attacks are to be determined. It monitors various LIDM as it may receive a message from AA of any LIDM whenever any attempt of read or write is made on it so that system administrators can take appropriate action. If an attacker is successful in crashing the AA of an LIDM located at say subnet j then GIDA can create the knowledge base of the crashed AA using the vulnerability database, subnet j security policy object repository (SUBNET J SPOR), subnet j configuration object repository (SUBNET J COR) and generating intrusion objects using the alerts of subnet j stored in its knowledge base to create intrusion object repository. After building the knowledge base it installs the crashed AA in the right subnet. This new AA will the create other components of LIDM to start monitoring of the concerned subnet.

One or more replica of GIDA can be produced for backup purpose and installed at various sites depending upon the network\organization specifications. Moreover to further prevent it from attacks we choose a mobile agent for this job, which changes location after random amount of time along with its replica. The advantage of having one or replica is also to handle coordinated attacks without requiring the agents to move from one location to another.

4. BEHAVIOUR OF PROPOSED SYSTEM IN DIFFERENT ATTACK SCENARIOS

The capabilities of the proposed system are understood from following examples. Scenario 4.1 describes how the components of the proposed system are installed and work in coordination to detect distributed attacks. Scenario 4.2 describes how the proposed system can be used to detect slow and coordinated attacks. In scenario 4.2 both LIDM and GIDA can be installed at one location and only LIDM can also be used if organization is not willing to protect AA of the LIDM.

4.1 Scenario 1: Detection of Distributed Attack

The company infrastructure is for this example is shown in Fig 8. Subnet 1 and subnet 2 are less secured so attacker is able to compromise one machine on each of them and subsequently install a bot [10] on each of them. Core subnet contains various servers for managing the network and storing critical information about the company. The attacker objective is to compromise the main server, which is controlling other servers in the subnet.

The analyzing agent in subnet1 generates alerts about multiple attempts done by attacker to compromise the machine. These included port scan alerts, several authentication failed messages, an authentication successful message and "su" privileges gained message. The alerts were sent to GIDA and stored in alert database of the subnet1. Similar alerts would be generated by analyzing agent of subnet2 and sent to GIDA. Core subnet will also generate alerts except of authentication successful and "su" privileges gained alert.

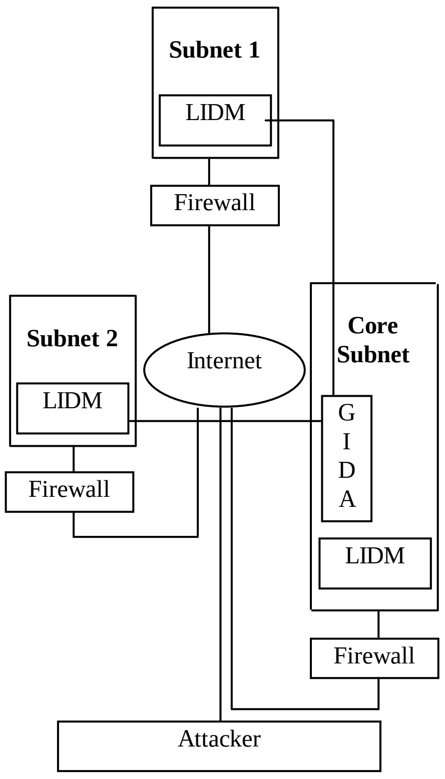

Fig. 8 Infrastructure Of Target Company

The CDCA of subnet1 will send data about multiple intrusion attempts access to CDAA of subnet2. Similarly CDCA of subnet2 will send its intrusion related data to core subnet's CDAA and CDCA of core subnet's would send its data to CDAA of subnet1. The CDAA of subnet2 will analyze data collected from CDCA of subnet1 and will generate approximate security level of subnet1. Similar analysis would be done for data collected CDCA of subnet1 and core subnet.

Due to the fact that attacker has compromised one machine in subnet2 and tried to compromise one machine in core subnet. The analyzing agent of core subnet suspects a distributed attack and looks for alerts in global database of GIDA to find attacks on other subnets. After finding that one machine of subnet1 has also been comprised and because attacks on all three subnets have occurred at the same time, it raises an alert of distributed attack and adds this alert in its distributed alert repository.

4.2 Scenario 2: Detection of Slow and Distributed Attack

The scenario is developed from the example of leading financial security company whose customer account information was exposed through a breed of attacks on web services done via SOAP requests. The report was published in leading newspapers in January 2005[4]. The company infrastructure and attack scenario is shown in Fig 9.

On day 1 attacker launched a port scan via drone1, which was detected by corporate firewall. On the basis of log information sent by corporate firewall the proposed system created an intrusion object and added this to source container object. Next several attacks launched at web application level via drone2 were logged by web server and application firewall. These logs were sent to proposed system, a new intrusion object was created corresponding to each attack and added to source container object. Moreover the source container object would be destroyed after copying its objects in a new target container object. This is done because initially system creates a source container type of object upon receiving the first log but later as in this intrusion attempts had a common target and not source so changing the type of container would be required.

After its creation target container object would be added to the queue of active target container objects. When no further intrusion attempts were

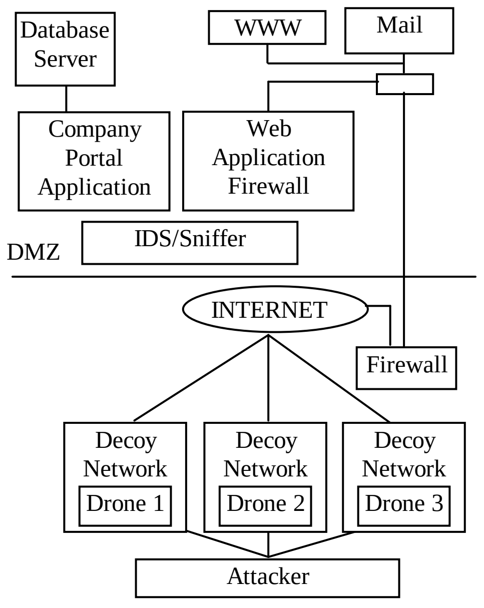

Fig. 9 Infrastructure Of Target Company

reported, this new object would be analyzed to determine whether the target system is vulnerable to intrusion attempts (represented by intrusion objects in the container object) indicated by container object. If on analysis target system is found vulnerable then an alert is raised otherwise no alerts are raised. Finally the target object is stored in intrusion objects repository for further reference. In this case since web server was completely locked and the application firewall was configured well to filter content based attacks so no alerts were raised and target container object was stored in intrusion object repository for further references.

On day 2 attacker performed various intrusion attempts via drone 3. The attacker activity and log where the activity was recorded is as follows: -

- Web application was crawled to download all links of web site and this was recorded nowhere
- Successful requests for different wsdl files were made and these were logged by web server.
- Parameter tampering was attempted via SOAP message to get critical information. These attempts were logged by web server.
- Execution of xp_cmdshell stored procedure to enumerate directory was recorded by database server.
- A text file was written to c:\inetpub\wwwroot folder and this event was recorded by operating system.

An intrusion object corresponding to each activity done by drone3 except for web crawling would be created by the proposed system on the basis of log information sent to it via different sensors and added to a target container object. Moreover a target container object matching the target of this new object would be searched in the intrusion object repository. As in this case if a target is found it is analyzed with this new object to identify any relationship between them. In this case there is no relationship but since various attempts to compromise portal were done on two consecutive days so both objects would be used to generate an alert. Finally this alert is stored in global intrusion database.

This shows the capability of the system to detect slow and coordinated attacks where multiple sources are hitting the target over a span of two days.

5. CONCLUSION

The proposed system can detect simple and complex intrusions by monitoring various system tools installed on a network or a host. As demonstrated in examples the system is capable of not only detecting simple distributed attacks but also the attacks which are slow, distributed and multi event based attacks normally not detected by many existing NIDS. Each agent used in the system performs a separate task and is independent. The system itself is robust and secure as each of its components is monitored by a different component, which makes it resist subversion. The system is both scalable and configurable as per the network requirements.

6. REFERENCES

[1] Przemyslaw Kazeinko, and Piotr Dorosz (April 2003). Intrusion Detection Systems Part I- (network intrusions; attack symptoms; ids tasks; and ids architecture.http://www.windowsecurity.com/articles/Intrusion_detection_Systems_IDS_PartI_network_intusions_attack_symptoms_IDS_tasks_and_IDS_architecture.html.

[2] Michael Karwaski. 2009. Efficiently Deducing IDS False Positives Using System Profiling. SANS Institute InfoSec Reading Room.

[3] Rafael Paez, and Miguel Torres. 2009. Laocoonte: An Agent Based Intrusion Detection System. International Symposium On Collaborative Technologies And Systems, pp. 217-224.

[4] Shreeraj Shah. 2008.Hacking Web Services. Second Indian Reprint. Delmar Cengage Learning, pp. 95-105.

[5] Amir Vahid Dastjerdi, Kamalrulnizam Abu Bakar. 2008. A Novel Mobile Agent Based Distributed Intrusion Detection System. World Acdemy of Scienec. Engineering and Technology 45 2008.

[6] Mohamad Eid, Hassan Artail, Ayman Kayssi, Ali Chehab. March 2008.LAMAIDS-A Lightweight Adaptive Mobile Agent Based Intrusion Detection System. International Journal of Network Security. Vol. 6, No.2, pp. 145-157.

[7] Abdoul Karim Ganame, Julien Bourgeois, Renaud Bidou, Francois Spies. 2008. A Global Security Architecture For Intrusion Detection On Computer Networks, Computers And Security. Vol. 27, pp.30-47.

[8] H.Q. Wang, Z.Q.Wang, Q. Zhao, G.F. Wang, R.J. Zheng, D.X. Liu. 2006. Mobile Agents For Network Intrusion Resistance; Springer. LNCS. vol. 3842/2006, pp. 965-970.

[9] Xiaodong Zhu, Zhiqiu Huang, Hang Zhou. August 2006. Design of a Multi-Agent Based intelligent Intrusion Detection System. First International Symposium on Pervasive Computing and Applications, pp. 290-295.

[10] Nicholas Ianelli, Aaron Hackworth. December 1, 2005. Botnets as a vehicle for online crime. CERT Coordination Center.

[11] Mohamad Eid, Hassan Artail, Ayman Kayssi, Ali Chehab. November 2005. Trends in Mobile Agent Application, Journal of Research And Practice In Information Technology. Vol.37, no.4.

[12] Ahmed Shaaban Abdel Shah, Imane Aly Saroit Ismail, S.H.Ahmed. December 2005. IDSUDA-An Intrusion Detection System Using Distributed Agents. CNIR Journal. Vol. 5, no. 1, pp.1-11.

[13] Richard A. Wasniwoski. September 23-24, 2005. MSABIDS-Multi-Sensor Based Agent-Based Intrusion Detection System. Information Security Curriculum Development Conference, pp. 100-103.

[14] Cheung-Leung Lui, Tak-Chung Fu, Ting-Lee Cheung. July 2005. Agent-Based Network Intrusion Detection System, in Proc. of the Third International Conference on Information Technology and Applications. Vol.1, pp.131-136, Sydney.

[15] K. Deeter, K. Singh, S. Wilson, L. Filipozzi, S. Wong. 2004.APHIDS- A Mobile Agent Based Programmable Hybrid Intrusion Detection. springer LNCS. vol. 3284/2004, pp.244-253.

[16] Mohamad Eid, Hassan Artail, Ayman Kayssi, Ali Chehab. October 2004. An Adaptive Intrusion Detection And Defense System Based On Mobile Agents, in Proceedings of the innovations in information technology. Dubai, UAE.

[17] Guy Helmer, Johnny S. K. Wong, Vasant Hanover, Les Miller, Yanxin Wang. 2003. Lightweight Agents For Intrusion Detection, The Journal Of Systems And Software. Vol. 67,pp. 109-122.

[18] A. F. Barika, N. El-Kadhi. November 2003. Intelligent and Mobile Agent For Intrusion Detection System. Proceedings of international conference of information and communication technology.

[19] P. Ning, S. Jajodia, X.S. Wang. September 2002. Design and implementation of a decentralized prototype system for detecting distributed attacks. Compu Communication. vol 25, issue 15, pp. 1374-1391.

[20] C. Kruegel, T. Toth. 2001. Applying Mobile Agent Technology to Intrusion Detection. Procedings of the ICSE for workshop on software engineering and mobility. pp. 1841-2002.

[21] G. Vigna, R. A. Kemmerer. 1999. NetSTAT: a network based intrusion detection system. Journal of Computer Security. Vol. 7, no. 1, pp. 37-71.

[22] S.R. Snapp, J. Bretano, G. V. Diaz, T. L. Goan, T. L. Heberlein, C. Ho, K. N. Levitt, B. Mukherjee, S.E. Smaha, T. Grance, D. M. Teal, D. Mansur. October 1999. DIDS (Distributed Intrusion Detection System)-motivation architecture and an early prototype. Proceedings of the 14th national computer security conference. pp. 167-176, Washington, DC.

[23] S. Staniford-Chen, S. B. Tung, D. Schnackenberg. Oct 1998. The Common Intrusion Detection Framework (CIDF). Proceedings of the Information Survivability Workshop. Orlando FL.

[24] Jai Sundar Balasubramaniyan, Jose Omar Garcia-Fernandez, David Isacoff, Eugene Spafford and Diego Zamboni. DC December 1998. An Architecture For Intrusion Detection Using Autonomous Agents, 14th IEEE Computer Security Applications Conference ACSAC. pp. 13-24, Washington.

[25] P. A. Porras and V. G. Neumann. 1997. EMERALD: event monitoring enabling response to anomalous live disturbances. Proceedings of the 20th National Information Security Conference, NIST. pp. 353-365.

[26] R. A. Kemmerer. 1997. NSTAT: a model based real-time intrusion detection system. Technical report TRCS97-18, Reliable Software Group, Department Of Computer Science, University of California at Santa Barbara, CA, USA.

[27] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yipi, D. Z. Erkle. 1996. GriDS: a large scale intrusion detection system for large networks. Proceedings of the 19th national information security conference. pp. 361-370.

[28] M. Crosbie and G. Spafford. Feb 1995. Active Defense Of A Computer System Using Autonomous Agents. Technical Report 95-008, COAST GROUP, Department Of Computer Sciences, Perdue University, West Lafayette, IN 47907-1398.

[29] A. Moijini , B. L. Charlier, D. Zampunieris, N. Habra. 1995. Distributed Audit Trail Analysis. Proceedings of the ISOC 95 Symposium on Network and Distributed System Security. pp. 102-112.

[30] B. Mukherjee, T .L. Heberlein, and K. N. Levitt. May/June 1994. Network Intrusion Detection. IEEE Network Magazine. Vol. 8, no. 3, pp.26-41.

[31] D. E. Denning. Feb. 1987. A Intrusion-Detection Model. IEEE Transactions on software Engineering. Vol. 13, no. 2, pp. 222-232.

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