A Simulated Multiagent-Based Architecture for Intrusion Detection System

Onashoga, S. Adebukola, Ajayi, O. Bamidele and Akinwale, A. Taofik
Department of Computer Science,
Federal University of Agriculture, Abeokuta Ogun State Nigeria.

***ABSTRACT*—In this work, a Multiagent-based architecture for Intrusion Detection System (MIDS) is proposed to overcome the shortcoming of current Mobile Agent-based Intrusion Detection System. MIDS is divided into three major phases namely: Data gathering, Detection and the Response phases. The data gathering stage involves data collection based on the features in the distributed system and profiling. The data collection components are distributed on both host and network. Closed Pattern Mining (CPM) algorithm is introduced for profiling users' activities in network database. The CPM algorithm is built on the concept of Frequent Pattern-growth algorithm by mining a prefix-tree called CPM-tree, which contains only the closed itemsets and its associated support count. According to the administrator's specified thresholds, CPM-tree maintains only closed patterns online and incrementally outputs the current closed frequent pattern of users' activities in real time. MIDS makes use of mobile and static agents to carry out the functions of intrusion detection. Each of these agents is built with rule-based reasoning to autonomously detect intrusions. Java 1.1.8 is chosen as the implementation language and IBM's Java based mobile agent framework, Aglet 1.0.3 as the platform for running the mobile and static agents. In order to test the robustness of the system, a real-time simulation is carried out on University of Agriculture, Abeokuta (UNAAB) network dataset and the results showed an accuracy of 99.94%, False Positive Rate (FPR) of 0.13% and False Negative Rate (FNR) of 0.04%. This shows an improved performance of MIDS when compared with other known MA-IDSs.**

Keywords- MIDS; CPM; Pattern-growth; Profiling

I. INTRODUCTION

Computer networks, including the Internet, have grown in both size and complexity. The services they offer made them the main means to exchange data and optimal environment for e-businesses. They have consequently become the means to network attacks. Intrusion detection technology provides reasonable supplement to the intrusion prevention systems such as firewalls, audit trails, system log etc. It has been a research focus for more than two decades from the publication of John Anderson's paper in 1980 (Anderson, 1980). Intrusion Detection Systems (IDSs) detect some set of intrusions and execute some predetermined actions when an intrusion is detected (Wang, 2006). Intrusion Detection has been achieved by following two different strategies of analysis.

- Anomaly detection: relies on models of "normal" behaviour of a computer system. Behaviour profiles maybe focused on users, applications or networks. Anomaly detection compares the defined profiles against the actual usage patterns to detect "abnormal" activity patterns. These patterns will be considered as intrusions.

- Policy detection: relies on a set of attack descriptions called attack signatures (Sasikumar & Manjula, 2011).

A Distributed IDS consists of multiple intrusion detection systems over a large network, all of which communicate with each other, or with a central server that facilitates advanced network monitoring, incident analysis, and inside attack data. Wrapping each of the components in IDS as mobile agent is aimed at effective intrusion detection in distributed environment. Intrusion detection in distributed environment requires data gathering, analysis and detection at every unit of the network and it has been established that mobile agent would perform well with this task (Sodiya, 2006). Agent is an entity being able to accomplish some work without manual intervention and supervision in certain condition and is able to migrate from host to host on a network under its own control. The agent chooses when and to where it will migrate and may interrupt its own execution and continue elsewhere on the network. An agent could also be static, that is, resides permanently on a platform performing one task or the other. In a distributed IDS system, each agent shares its data with other agents in the system (Oriola et. al, 2012).

Mobile Agents are considered to be an effective choice for many applications for several reasons. The ability for mobile agents to sense their environments and react dynamically to changes is useful especially in intrusion detection. As mobile agents, the IDS can evade attacks. The constant movement of these agents in a network among multiple hosts makes it difficult for an attacker to locate and disable them. A Multiagent system is a system in which several interacting, intelligent agents pursue some sets of goal, or perform some sets of tasks. Multiple Agent system can adopt the characteristic of mobility to carry out activities in a flexible and intelligent manner that is responsive to changes in the environment (Bradshaw, 1997). It consists of a number of agents, which interact with one another, typically by exchanging messages through some computer network infrastructure (Wooldridge, 2002).

This paper makes an attempt to propose solutions to having a better detection rate and to actualize the real implementation of a Multiagent system for intrusion detection in an environment by introducing an intelligent and flexible MultiAgent archtitecture for IDS (MIDS). MIDS is designed based on the following interests:

a) Improving time-to-detection of MA-IDS.

b) Provide an architecture where real time attacks are efficiently detected.

c) Aims at reducing effectively, false alarm rates mining the closed frequent patterns of users' activities (Onashoga, et. al, 2009) for profiling.

d) Since the role of IDS is to monitor and ensure security of the network, the MA-IDS itself is a primary target of attacks. It then becomes important for IDS agents to operate in hostile environment and still exhibits a high degree of fault-tolerance and performance.

The rest of the paper is organized as follows: Section 2 reviews past researches that have been done in the area of mobile agent based intrusion detection system. Section 3 details on the proposed architecture and its considerations for achieving a better detection rate, while section 4 describes the testbed implementation procedures and its performance analysis. Section 5 concludes the work.

II. Existing Related Researches

Herrero & Corchado (2009), Kamaruzaman et. al. (2011) and Onashoga et. al. (2009) reviewed some related work on MA-IDSs with focus on the tasks, architecture and implementation of agents. In contrast, an extensive critical review of MA-IDS with focus on their architecture, techniques, strengths and weaknesses is done in this paper. This section details the reviews of recent related researches.

Sasikumar & Manjula (2012) proposed a 3-layer fault tolerant architecture. The architecture consists of Host and Net agents at the first layer, the mobile agents at the second layer and the Decision-making and Replication agents in layer 3. The Host Agent's function is to protect the host. When suspicious activities can't be decided, the Host Agent generates an Intrusion Detection event and transmits it to layer2. The role of the Net Agent is to detect network intrusions. It supervises the network traffic, records all suspected events in a data base and responds intrusions. It also installs mobile agent platform on it. The mobile agent is responsible for collecting information of an attack from the Host Agents or Net Agents for further analysis in layer3. The Decision-making Agent analyses the data collected by the mobile agent and passes the control to Replication agent, who in turn is responsible for replication and recovery management. The fault tolerance reported is the data analysis by agent and the pass of control to the replication agent. The techniques used is not reported. Also the security of the agents is not considered.

Oriola et. al. (2012) proposed a peer to peer architecture that integrates the concept of multiagent system and data mining. The architecture proposed has 3 levels namely: The first level which is the core of the system. It is at this level that the interaction and integration of static and decentralized multi-agent system and distributed data mining is established. The second level is made up of dedicated and specialized agents that cooperate and communicate to generate host based and network based intrusion detection system. At the highest level of composition, different intrusion detection systems are involved. Their mode of cooperation, communication and detection capability are not reported..

Zeng and Duo (2009) followed a typical network-based application where there are more than one application servers, database servers and clients. The model designed consists of three agents namely: client agent, communication agent and server agent. Client agents are installed on a client workstation, and responsible for collecting extra user information and then send to server agents with the help of communication agents. Server agents run in the server where masquerade intrusion is to be detected. They process the message sent from client agents, read from and write to user model, server agents can make a decision on whether the current user is a legal one or not according to a predictive model. Communication agents monitor the client agent's request. After the received message is parsed, the useful message is forwarded to the server agents. The client agents collect user information such as operating system, log files, network card etc. Zeng and Duo (2009) adapted the use of Hidden Markov Model (HMM) to model user's activities on server. The algorithm makes sure that the likelihood of sequence with respect to the HMM increases after each iteration in the training. These training sets are constructed from the event database. However, a data mining algorithm is used to filter the events that seem abnormal, because abnormal user sequences are not allowed to be included in the training set. The strengths of this algorithm lie on (1) the focus of the model on detecting only one type of intrusion – masquerade attack (2) the model makes use of real time detection. The security of the agents is not reported.

DNIDS architecture proposed by Kuang (2007) consists of 5 components namely Sensors, Detectors, Alert Agents (AA), Maintenance Agents (MA), the Manager, and the Console. Sensors capture the network packets from a network segment and transform them into collection-based vectors. The Detector is a collection of CSI-KNN (Combined Strangeness and Isolation K-Nearest Neighbor) classifiers that analyze the vectors supplied by the sensors. The 3 agents are only designed for intrusion tolerance not for intrusion detection and are only installed on the administrative server. Detection rate of the architecture proposed is not reported in this work.

Abraham et.al (2007) proposed a hierarchical architecture with Central Analyzer and Controller (CAC) as the heart and soul of the DIDS. The CAC usually consists of a database and webserver which allows interactive querying by the network administrator for attack information/analysis and initiate precautionary measures. CAC also performs attack aggregation, building statistics, identify attack patterns and perform rudimentary incident analysis. The mode of data collection is not discussed but the algorithm is tested on the KDD cup 1999 dataset whose source is network based. The authors tested the model using different soft computing techniques which consists of neural network, fuzzy inference system, approximate reasoning and derivative free optimization techniques on a KDD cup dataset. The experiments have three phases namely: input feature reduction, training phase and testing phase. In the data reduction phase, important variables for real-time intrusion detection are selected by feature selection. In the training

phase, the different soft computing models are constructed using the Labeled data. The test data is then passed through the saved trained model to detect intrusions in the testing phase. The problem faced with hierarchical architecture is being solved by allowing a free communication between the layers. A well comparative analysis of the different soft computing algorithms with other machine learning techniques is being carried out which serves as references for researchers in the field. The full description of how the agents detect intrusions based on the soft computing algorithms proposed is not well discussed.

Sodiya (2006) proposed a two-level architecture coined MSAIDS. The first level is the Lower Level Detection (LLD), which has the data agents and processing agents. The data agents move around the nodes in the network to collect associated information. The 2 processing agents also known as node agents where Node-1 agent is responsible for construction of the first level database from the information collected and for data cleansing, classification and formatting. The Node-2 agent is responsible for data mining and first level intrusion detection and communicates the possibility of intrusions to the interface agent through the alarm agent. The Upper Level Detection (ULD) also known as confirmation level is involved in separate intrusion detection process. At the ULD, the lower level agents gather data from the data agents and inform the Controller and Protector (CP), which acts as the Facilitator agent about the nature of the data gathered. The CP also ensures proper communication and delivery of service among agents. The data gathered are then used to update the ULD database; the ULD does not check for intrusion if there is no signal from the LLD. The types of data collected are application messages, authentication events, system calls, TCP connections. An Apriori algorithm is modified to extract patterns by the first level and second level agents. MSAIDS maintains security of agents by using asymmetric cryptosystem of the Aglet's framework. In addition to this, agents' states are recorded and authenticated before they are initiated. Any suspected intrusion is reported by the Interface Agent to the Site Security Officer (SSO). The action to be taken by the SSO is not stated. In addition to securing mobile agents, the use of recorded state mechanism, which has been proved effective, is a plus in this work. The drawbacks identified are firstly, the activities at the ULD could still be integrated with the LLD to form one-level architecture and have the CP atthe ULD since detection of intrusion at each level is still based on same algorithm. It took 0.14 seconds to report an intrusion at the LLD and 0.75 seconds at the ULD. And secondly, the architecture presented does not provide security for the database, which could be vulnerable to changes by attackers.

Wang et. al. (2006) designed a system framework which includes the Manager, who is the centre of controlling and adjusting other components. It maintains their configuration information. The manager receives intrusion alarms from host monitor mobile agent and executes intrusion responses using intrusion response mobile agent. It also consists the host monitor mobile agent that resides on every host in the network. If intrusions occur confirmatively, the host monitor MA will appeal to the manager and report the suspicious activity directly. After receiving the appeal, the manager distributes a data gathering MA patrolling other hosts in the network to gather information. If a distributed intrusion is found, the manager will assign an intrusion response MA to respond intelligently to every monitored host. The database of configuration stores the node configuration of detecting system. The data source of IDS is both host-based and network-based. The gathering part of data source is to record, filter and format the connection information of the monitored host and write them into the log. The types of data collected includes system log and some conserved audit records. The intrusion analysis mobile agent mainly analyses the log file in the monitored host system and compares them with the characters of known attack activities to find abnormal activity combined with different detection measures which were not mentioned. The framework's security is based on the security measures provided by Aglet. The intrusion response MA responds to the intrusion events that occur which can include tracking the intrusion trace to find the intrusion fountain, recording the intrusion events into database etc. It changes the hierarchical system structure of traditional distributed IDS. The major drawback lies at the control centre carrying out the major part of the intrusion detection, if the location of this centre is discovered, then the system collapses.

III. PROPOSED ARCHITECTURE

This section presents a full description of Multi-agent based Intrusion Detection System (MIDS). MIDS's agents' architecture consists of both static and mobile agents. MIDS uses an algorithm named Closed Pattern Mining (CPM), which adopts a data mining descriptive model for user profiling. This is now followed by a full description of the components of MIDS.

A. Mids Architecture

The MIDS architecture, shown in Figure 1 adopts the data mining algorithm (CPM) for user profiling. It adopts a real time mode of detection in a network environment. MIDS architecture structure is into two parts:

1) *The IDS Control: resides on every host;*
2) *The Administrative Control: resides on the server.*

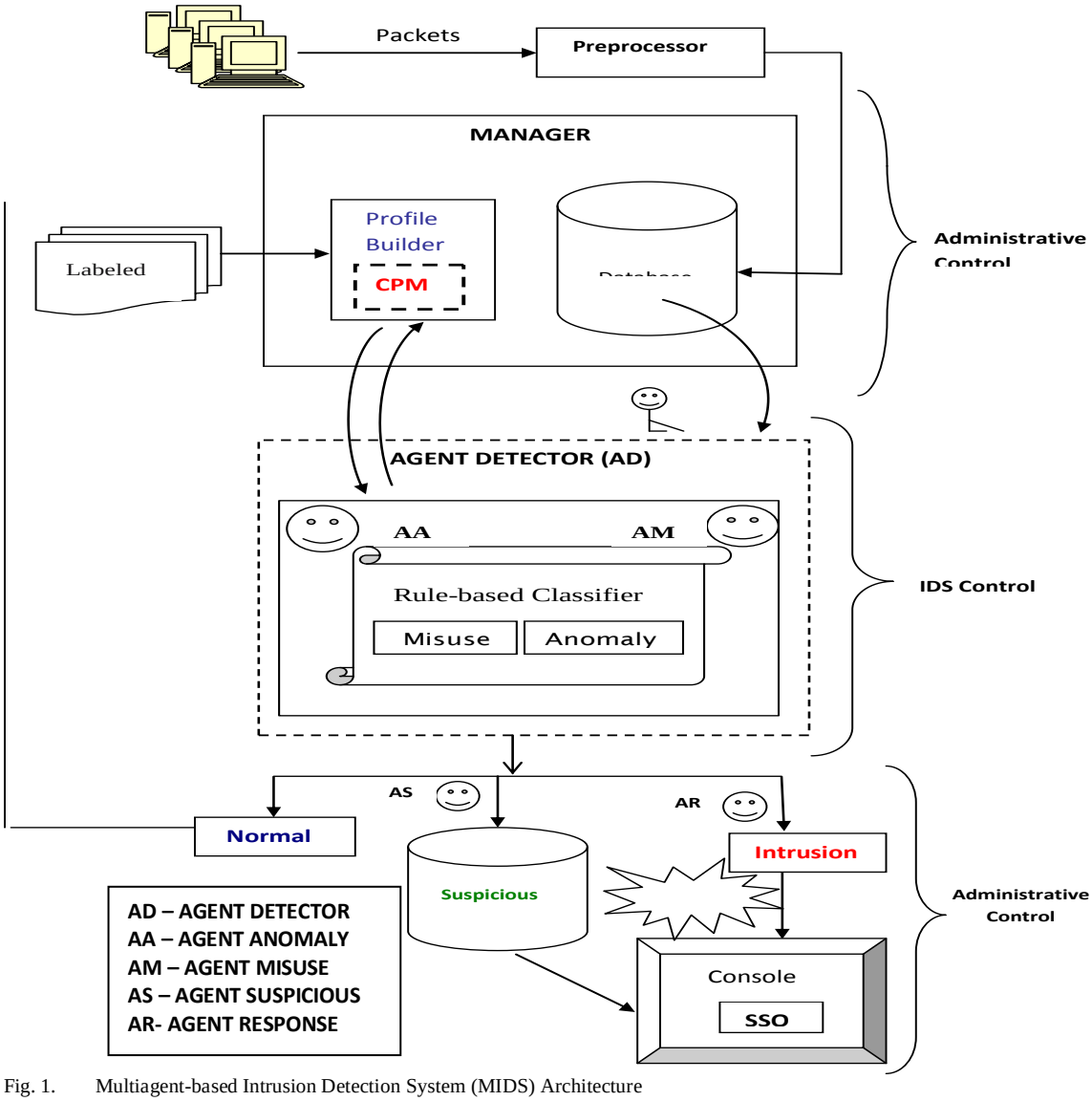

Fig. 1. Multiagent-based Intrusion Detection System (MIDS) Architecture

In general, the architecture consists of 3 major phases which involve the data gathering phase, the Detection phase and then the Response phase which is passed to the Site Security Officer (SSO).

Each of these phases is explained as below:

3.1.1 Data Gathering phase

The data gathering phase involves collection of data and profiling stage. The data collection is done on both the host and network. Each of the host has a sensor, a network sniffer is integrated in the sensor and is used to gather all network packets. As data streams travel back and forth over the network, the sniffer captures each packet and eventually decodes and analyzes its contents according to specifications. For example, *tcpdump*, a network debugging tool can serve as a sniffer and monitor the packets transmitted over a network. The preprocessor is responsible for accepting raw packet data and producing records. This component is capable of reading packets from the tcpdump file. The output produced by this component consists of records which are now stored in a database. Record contains aggregate information for a group of packets.

In MIDS architecture, the roaming agent (RA) consists of three parts: the code, itinerary and results. It moves from host to host to gather data following a predefined itinerary established by the Manager. The Manager acts as the supervisor of all agents in the architecture. Each record gathered is now stored in the database. The profiling stage involves the use of CPM algorithm by mining only the closed patterns of users' activities.

3.1.2 Detection phase

The architecture makes use of the two approaches for detection: Anomaly and Misuse approach. This phase is the IDS Control part of the architecture. At this phase, the Agent Detector collects the newly arrived record and clones (using the clone() method) itself into "two": the Agent Misuse (AM) and Agent Anomaly (AA). The AA, who is in charge of anomaly detection takes this record and checks the Profile Builder (where CPM resides) for the user's profile. This is

done so as to know whether the behaviour is in line or deviates from the normal user's profile. On the other hand, the AM searches for any attack signature in the record based on the rules defined to check for an attack. See5 tool (RuleQuest, 2007) is used for rule classification. See5 is a GUI based software which is capable of classifying large volumes of data within a second depending on the speed and specification of computer processor. The rule-based classifier is used in this work so as to have an idea of how rules are being defined for misuse detection. The following are excerpts of the rules which the cloned agents now classifies the event as "Normal", "Suspicious", or "Intrusion":

a) Suspicious

An event is said to be suspicious if it slightly deviates from the normal behaviour of the user, that is, its item falls within a particular range, for example:

```
if (hot<3 || (failedLogin <=5 | srcByte <=100 | destByte <= 50) && service == "hotspot")
        {
                System.out.println ("DETECTOR >> User: ["+id+"] Suspicious!");
                msg.sendReply(1);
        }
```

This event is now transferred to the Agent Suspicious (AS) which first stores this in a database for monitoring and then passes it to the SSO through the console.

b) Intrusion

Some rules are also defined for categorizing intrusive events e.g. an event is said to be "intrusion" if it deviates entirely from the normal behaviour, that is, the values of its items passes the ones considered for "suspicious" or some discrete values are not seen. For example, in a system where the only network service used is "hotspot", any event that has a service aside this is taken as "intrusion".

c) Normal

Any other event outside the conditions in (a) and (b) is considered "Normal". The Agent Detector is responsible for passing the "Normal" user to the Profile Builder for update. The condition for classification of normal records is as shown in the example below:

```
if (hot==0 || (failedLogin <=3 && srcByte <=128 && destByte <=64) && service == "hotspot")
        {
                System.out.println("DETECTOR >> User: ["+id+"] Normal!");
                msg.sendReply(1);
}
```

3.1.2.1 CPM Algorithm

This section first describes the factors behind the development of CPM algorithm and then details the pseudocode of the algorithm. The Closed Pattern Mining (CPM) adopts the concept of FP-growth algorithm due to its advantages, by mining a prefix-tree called CPM-tree. CPM performs the closure checking on the fly with only one scan over the dataset unlike in FP-growth algorithm where two scans are needed.

CPM tree is proposed to perform the closure checking of any real time intrusion. The CPM tree contains only the closed itemsets and its associated support count, unlike the FP-tree which contains all the information of the database.

The main steps of the CPM algorithm are as follows:

1) Perform a closure checking on the first specified number of transaction, T. (note that the number of transactions depends on some selected features).*

2) Construct the CPM tree with each node containing only the closed itemsets and its associated support count which is a compact representation of the database (Note: every user has a node in the CPM tree).

3) When a new transaction arrives, the CPM algorithm checks whether itemset, X □ T is in the current closed itemsets for a particular user. If it is, it updates X's support, otherwise, if X is a newly arrived closed itemset, the algorithm adds it as link to the existing node of that user in the CPM tree.

4) A transaction is deleted if within a time window, a particular transaction is not frequent. Then the node is pruned out in order to reduce the memory usage of the CPM tree.

5) The closed frequent itemsets (transaction) can be output any time at the Site Security Officer's request by traversing the CPM tree.

The CPM tree is used to maintain the current closed itemsets. Each node in the CPM tree stores a closed itemset, its current support information, and the links to its immediate parent and children nodes.

Pseudocode of CPM algorithm is illustrated below:

Algorithm 1: CPM – Addition

```
X_close = true; Cnew = φ;
 procedure Add(X, C, Cnew)
          if (X ϵ C)
                   for all (Y ⊆ X and Y ϵ C)
                            Y_s ← support(Y, C) + 1;
                    end for
          if (X_close = true) return;
 else
          if (support(X, C) > 0 )
                   if (Cnew = φ )
                            X_0  ←  X;
                            C_new  ←  X;
                            X_close = false;
                            X_s  ←  support (X, C) + 1;
                   else
          Xc = φ;
                   for all ( K ⊃ X and K ∈ C)
                                     if (len(K) < len(M) then
                            M = K;
                   end for
                            X_c ← M;
```

```
                                    if ((X_c / X) ∩ X_0 = φ and
                                    X_c ≠ φ )
                                     C_new ← C_new ∪ X;
                                     Xs ← support(X, C) + 1;
                                    end if
                                    end if
                                    else
                        if (C_new = φ ) then
                                     X_0 ← X;
                                     C_new ← X;
                                     X_s = 1;
                        end if
                        end if
                         end if
              for all (m ⊂ X and Len(M) = Len(X) -1
                        call Add (M, C, C_new);
             end for
                        if (X = X_0)
                                  C ← C ∪ C_new;
                                  Support (X, C) = X_s;
                        end if
end procedure
```

3.1.2.1.1. Illustration of CPM algorithm

As an illustration, this part describes the use of closed frequent itemsets to construct anomaly detection. In profiling users' behaviour, the normal users are selected from the labeled data; and a user-id is the major key. Collect all the transactions for each user and find the closed patterns.

For example, the following transactions occur for users with *service, hot, failed login, src_bytes, dst_bytes, duration* as the 6 features picked respectively:

SP118 = {hotspot, 0, 1, 93, 15, 4:40 . . . }
SP118 = {hotspot, 0, 4, 13, 20, 3:15 . . . }
SP118 = {hotspot, 0, 0, 51, 41, 0:43 . . . }
SP172 = {hotspot, 0, 4, 26, 45, 8:55 . . . }
SP172 = {hotspot, 0, 2, 76, 52, 2:42 . . . }
SP144 = {hotspot, 0, 8, 71, 49, 6:20 . . . }
SP144 = {hotspot, 0, 4, 54, 2, 4:20 . . . }
SP144 = {hotspot, 0, 0, 92, 18, 6:81 . . . }

Applying CPM algorithm as detailed in section 3., with priority labels assigned to the features that could greatly contribute to having a normal activity e.g. *failed login, src_bytes, dst_bytes*. As an example, transaction *{hotspot, 0, 1, 93, 15, 4:40 . . . }* is stored in CPM tree for *SP118* being a superset of its first 2 transactions, giving a support of 2. The 3rd transaction is not left out as a transaction for the user.

Also the 2nd transaction for SP172 is closed and then stored in the CPM tree with support of 2.

Lastly, the 3rd of *SP144*'s transaction is considered close with support of 2, in comparison with the 1st transaction, not leaving out the 2nd transaction with support of 1.

The CPM tree constructed for this example is as shown in Figure 2 with the nodes identified by the user-id.

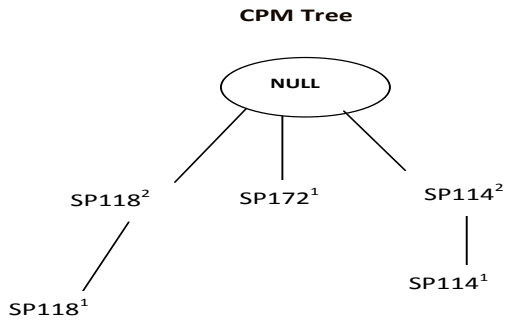

Fig. 2. The resulting Closed Pattern Mining (CPM) Tree for the illustration

3.2.2 Response phase

This phase is part of the Administrative control, but it could as well be on the host too. The Agent Response is responsible for alerting the SSO of every occurrence of intrusion based on real time module. The SSO now takes an active action which could be "Shut down the host", "Logout the user" e.t.c. Either AM or AA stores the "suspicious" events in a database for check in case of any reoccurrence (> 2), the attention of AR is then called to alert the SSO for an action to be taken.

Console

The Console is an interface that allows control of agents, management of the list of monitored systems, and intrusions are reported through it.

SSO

The Site Security Officer (SSO) is the network administrator who is in charge of the entire network environment and its resources. He/She takes an action when any alarm is raised.

Database

This is where all the data gathered are stored. It also contains the records of both the normal events and suspicious events detected by the agents in charge of that task. This is being put in check by the Manager.

The critical component is the Manager agent who resides on the server, MIDS is designed in such a way that if at all the Manager is compromised, which implies that no agent would have access to the profile or even the database, then the AA is made to keep tracks of all the user it has checked for. If a new user now visits a host, the AA calls the attention of the RA to check for that user's information on the other hosts on the network. The RA is encoded with checking rules to identify the failure of the detector.

B. Communication, Coordination and Security of MIDS' agents

All agents in this framework communicate and collaborate with peer agents, using a subset of the agent communication language and protocol, Knowledge Query and Manipulation Language (KQML). An agent's request for information or service is defined or encoded in KQML format and it is transported to the provider using ATP – Agents Transfer Protocol.

For the coordination and security of the agents in MIDS, on every host are the Static Agent (SA) and the IDS Control with its embedded agents. The Static agent (SA) ensures the security of the mobile agent platform. As part of the advantages of having SA on every host, is to stop the host from treating the agents as it likes, this problem is usually referred to as the *malicious host problem.* When an agent visits a host, the SA authenticates the agent before any interaction, when it is found to be from the right source it issues it a certificate, then it is allowed to perform its task. The notion of Digital Signature Algorithm is used in this case.

At the Administrative Control point, are the Manager, the Database and the Profile Builder. The Manager has the Agent Control list (ACL) which contains all information about the agents dispatched on the network; its movement, its identifiers and its state.

IV. Implementation Of MIDS

This section describes the implementation of MIDS architecture. Aglet Software Development Kit (ASDK) described in Lange and Oshima (1998) is used in this work. MIDS is implemented and tested over a UNAAB network dataset using java 1.1.8 and Aglet 1.0.3. The ASDK environment, developed by IBM provides mobility facilities to agent programs. It is written in Java, and includes primitives to create, move, communicate and dispose programs. A mobile agent in ASDK is known as an Aglet (contraction of agent + applet). The aglet migrates from one machine to another with the help of a server module, known as Aglets Server or Tahiti Server.

A. Testbed Implementation

In the test carried out, two hosts are established with Windows Vista operating system in a LAN to construct a distributed platform based on Aglets as earlier described. The first host acts as the monitor host while the second one is the monitored host, each of these hosts has the UNAAB dataset.

UNAAB network log, which is a dataset downloaded from the local network connections of the University of Agriculture, Abeokuta, contains the network activities of users over a period of time. The labeled data consists of 5 weeks log of users' activities of 212644 records, where 201,540 are labeled as "normal" and 11104 as "intrusion". CPM is used on the labeled data for profiling The features include ip-address, Protocol type, User-ID, Service, Hot, Scr_bytes, Dst_bytes, Failed_logins, Duration, Period, Label etc. There are two main attack types found in this dataset, hese are guess and sniff attacks.

Table 1 shows an example of network connection records in UNAAB network log with the features. Given a set of records, the CPM algorithm combined with the rules for classification can assign a label to describe each record in the unlabeled data.

The Tahiti server shown on Figure 3 offers a graphical user interface to run the agents described in this architecture and enables the loading of the MIDS's agents' classes for deployment. It thus provides the useful environment to run the implementation codes written in Java.

Once the Agent is loaded, the autonomous/mobility activities can be performed by the agent. In implementing MIDS, four classes are created: MIDSManager.java, MIDS_AD.java, MIDS_AA.java, MIDS_AM.java.

B. Performance Evaluation of MIDS

The result displayed in Figure 4. The small pop-up shown in red is the alert displayed on detecting a "suspicious" event. The other window at the background is the console showing the activities of the mobile agents on execution.

TABLE I. Examples of connections found in UNAAB network log

User-ID	**Service**	**hot**	**Src_bytes**	**Dst_byte**	**Failed_logins**	**. . .**
SP 862	Hotspot	0	100	62	0	. . .
Admin	Hotspot	2	35	76	1	. . .
SP499	ftp	0	78	32	5	. . .
JP1042	Hotspot	1	54	20	0	. . .
SP535	http	0	8	78	0	. . .
. . .	. . .	. . .	. . .	. . .	. . .	. . .

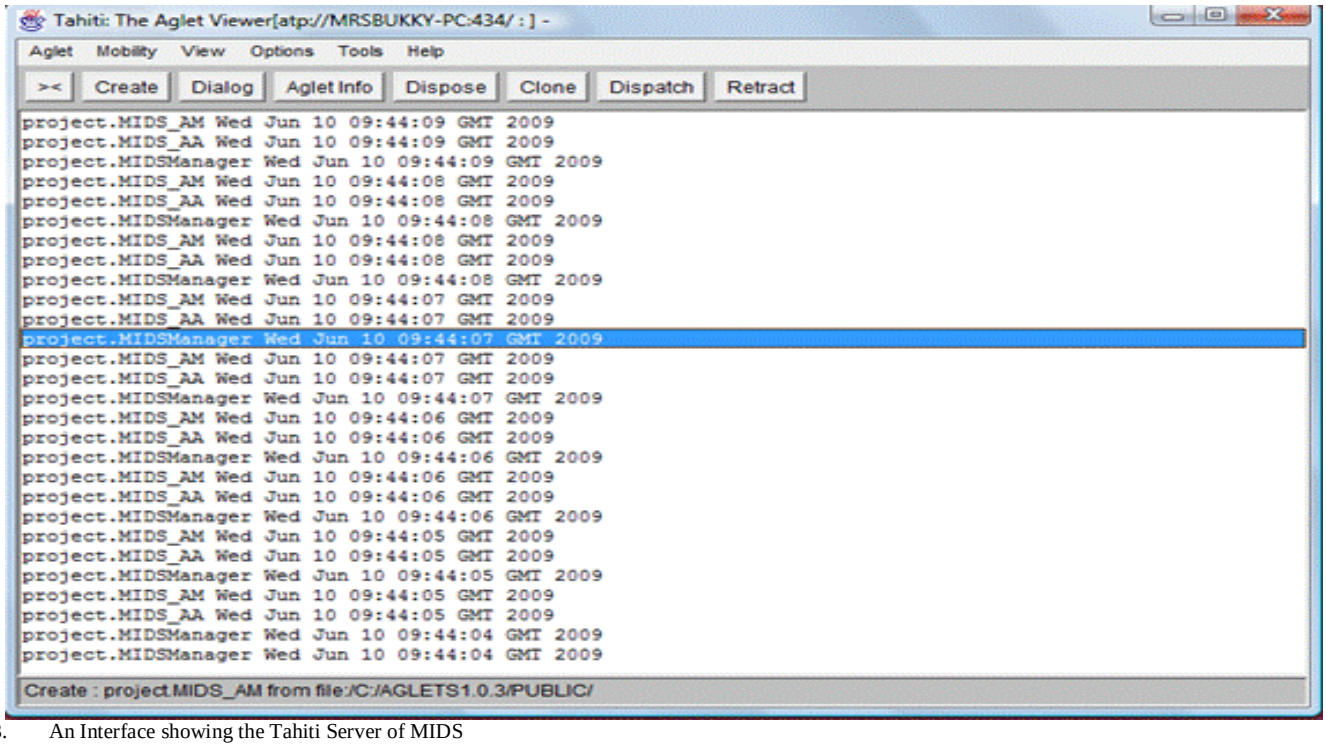

Fig. 3. An Interface showing the Tahiti Server of MIDS

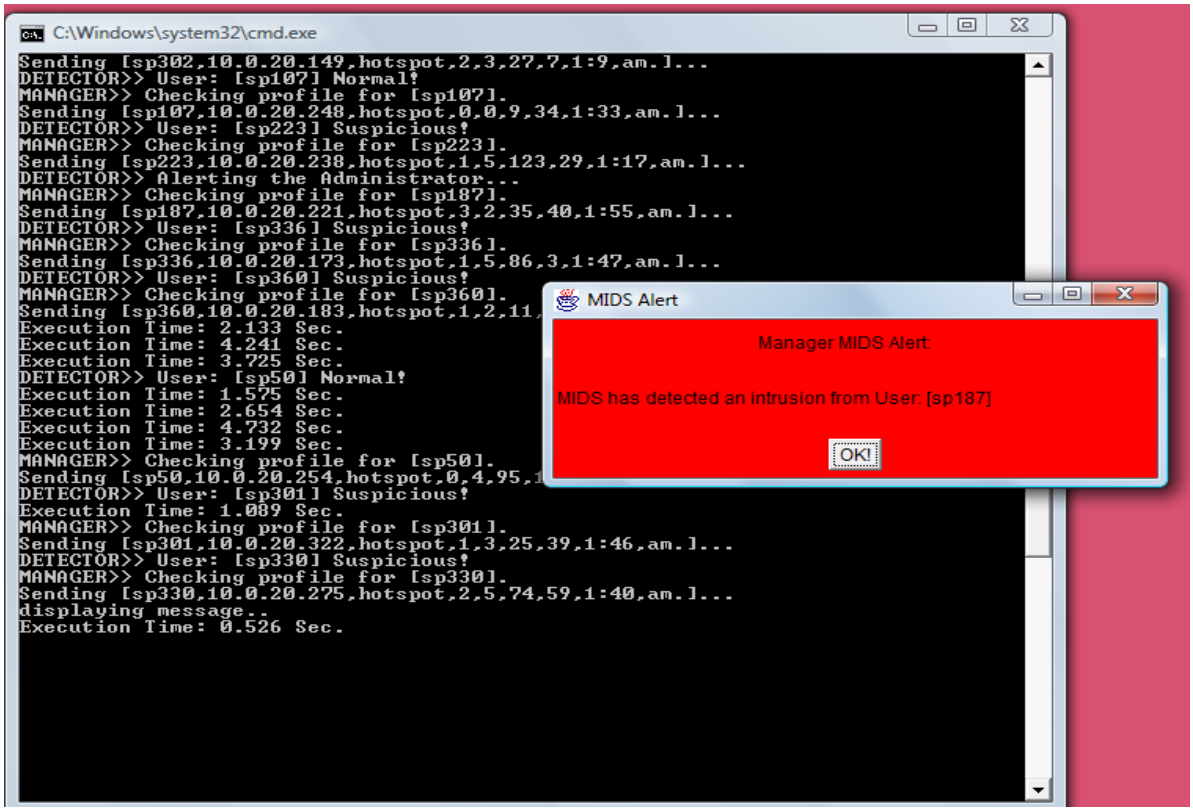

Fig. 4. Alert on detecting Intrusio

The records of the classification are taken by counts in order to identify the false alarm rates. The addition of the number of “suspicious” and “intrusions” are classified as “intrusion”. Table 2 shows the classification on the unlabeled data.

TABLE II. Results of MIDS on UNAAB network dataset

	Cases	**False Positive Rate**	**False Negative Rate**
Normal	201540	0.13%	99.87%
Intrusion	11104	99.96%	0.04%

Using the 4 metrics for evaluation of IDS, the following results are discovered: Accuracy = 99.94% ,

Detection rate = 99.96% ,False Positive Rate = 0.13%, and False Negative Rate = 0.04%

1) Timing Result

The performance in terms of time to detection and the mean time of reporting an intrusion are considered. This involves the system recording each time an intrusion is detected and then calculates the mean. It is found out that the mean time for reporting an intrusion is 0.67seconds. On the other hand, it took 0.89secs to report an intrusion in Sodiya (2006) while Eid (2004) recorded the overall trip for the agents in its architecture as 4.42 secs starting from activation of the sniffer to completion of the processing from one host to the other.

2) Performance Analysis

UNAAB dataset is divided into batches of 10%, 25%, 50%, 75%, 100%, the True Positive Rate and False Positive Rate are calculated as shown on Table 3 and depicted in Figure 5.

TABLE III. Comparison on batches of UNAAB network log

		BATCHES				
		10%	**25%**	**50%**	**75%**	**100%**
MIDS	**TPR (%)**	94.5	97.43	98.89	99.54	99.96
	FPR (%)	3.9	3.3	2.2	1.7	0.13

V. CONCLUSION

As network attacks become more alarming, exploiting systems faults and performing malicious actions, the need to provide effective intrusion detection methods increases. Distributed attacks are especially difficult to detect and require coordination among different intrusion detection components or systems. The idea of mobile and autonomous components intuitively seems useful in intrusion detection, hence the use of multiagent system. A data mining approach is provided in this paper to enhance the detection performance of the agents deployed in the design. Experiments performed emphasize the aim of applying agents to detect intrusions.

ACKNOWLEDGEMENT

Our appreciation to the ICTREC staff for giving us access to the University network log.

REFERENCES

[1] Anderson J. P. (1980). “Computer security threat monitoring and surveillance”, Technical report, James P. Anderson co, Box 42, Fort Washington, February 1980.

[2] Bradshaw, M. J. (1997). “An introduction to Software agents”, In Jeffrey M. Bradshaw, Editor, Software agents, Chapter 1, AAAI press, The MIT press,1997.

[3] Eid, M., Artail, H., Kayssi, A. and Chehab, A. (2004) “An Adaptive Intrusion Detection and Defense System based on Mobile Agents”, Proceedings of the Innovations in Information technologies (IIT’2004), Oct, 2004, Dubai, UAE.

[4] Kuang, L. V. (2007). “DNIDS: A Dependable Network Intrusion Detection System using the CSI-KNN”, M.Sc. thesis, School of Computing, Queen’s University,Ontario, Canada.

[5] Lange, D. B. and Oshima, M. (1998). “Programming and Deploying JavaTM Mobile Agents with Aglets”, Addison Wesley Longman, Inc.

[6] Oriola, O., Adeyemo, A. B. & Robert, A.B.C. (2012). "Distributed Intrusion Detection System Using P2P Agent Mining Scheme", IEEE African Journal of Computing & ICT, Vol 5. No. 2, March, 2012 ISSN 2006-1781

[7] Onashoga, S. A., Akinde, A. D. & Sodiya, A. S. (2009). "A Strategic Review of Existing Mobile Agent- Based Intrusion Detection Systems", Issues in Informing Science and Information Technology Volume 6, 2009

[8] RuleQuest, (2007). Retrieved from http://www.RuleQuest.com

[9] Sasikumar, R. & Manjula, D. (2011). "A Distributed Intrusion Detection System Based on Mobile Agents with Fault Tolerance", European Journal of Scientific Research, Vol.62 No.1 (2011), pp. 48-55

[10] Sodiya, A. S. (2006). “Multi-Level and Secured Agent-based Intrusion detection System”, Journal of Computing Science and Information Technology, vol. 14, no. 3.

[11] Jain, P., Raghuwanshi, S. and Pateria RK (2011) "New Mobile Agent-based Intrusion Detection Systems for Distributed Networks", International Journal of Wireless Communication Volume 1, Issue 1, 2011, pp-01-04

[12] Kamaruzaman, M., Shukran, M. A. M., Khairuddin, M. A. & Isa, M.R.M. (2011). "Mobile Agents in Intrusion Detection System: Review and Analysis". Modern Applied Science Vol. 5, No. 6; December 2011

[13] Wang, W., Behera, S. R., Wong, J., Helmer, G., Honavar, V., Miller, L., Lutz, R., and Slagel, M. (2006). “Towards the Automatic Generation of Mobile Agents for Distributed Intrusion Detection System”, Journal of Systems and Software, Vol 79, pp:1-14. www.elsevier.com/locate/jss.

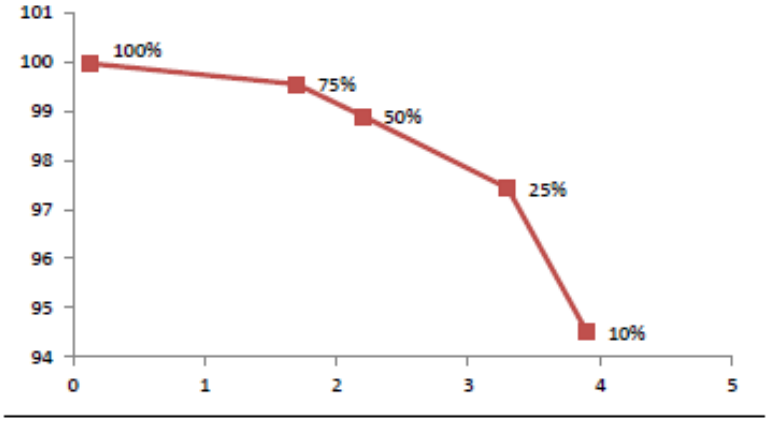

Fig. 5. Comparison of TPR (%) against FPR(%) on batches of UNAAB dataset