A Cloud-Based Model for Computer Network Threats Management.

I.K. Ogundoyin, Ph.D.[1*]; C.O. Akanbi, Ph.D.[1]; A.A. Adigun, Ph.D.[1]; and S.A. Akinboro, Ph.D.[2]

[1]Department of Information and Communication Technology, Faculty of Basic and Applied Sciences, College of Science, Engineering and Technology, Osun State University, Osogbo, Nigeria.
[2]Department of Computer Science, Bells University of Technology, Otta, Nigeria.

E-mail: ogundoyin@uniosun.edu.ng *
olaac@ymail.com

ABSTRACT

In this paper, a cloud model was developed to manage threats beyond users' networks capability. This is with a view to addressing threat database growth which degrades users' networks performance as a result of constant threat update, and lack of expertise to mitigate emerging threats. The model was designed using Unified Modeling Language (UML) and Artificial Neural Network (ANN) techniques. The model produced encouraging results and a great potentiality to enhance cyberspace protection.

(Key terms: threats, cloud, model, cyberspace, management, UML, unified modeling language, ANN, artificial neural network)

INTRODUCTION

The increasing usage of Internet for almost every aspect of life has greatly increased the Internet users' risk as a result of emerging threats and attacks to critical information infrastructure and computer networks which have grown in dimensionality and sophistication (Wamala, 2011). This is because the emerging cyber threats are complex, sophisticated, and detrimental to global cybersecurity and socio-economic survivability of any nation (Wamala, 2011). However, there have been research efforts to salvage the situation, among technologies produced as a result of research efforts in this line are conventional technologies like encryption system, firewall, antivirus, antispyware, intrusion detection system, and intrusion prevention system (Govindarajan and Chandraseckaran, 2010; 2011; Nabil *et al*., 2012; Ogundoyin *et al.,* 2013).

Most research efforts in area of threats management have been focused on improving threats detection accuracy, reducing false alarm rate, integration of the available threat defense technologies and novel threats information sharing using different techniques (Adnan and Michael, 2014; Mafra *et al.*, 2014; Bul'ajoul *et al.,* 2015; Gisung *et al.,* 2014; Jamal and Samuel, 2016; Bin and Jingbo, 2014; Kumar *et al.,* 2015). Unfortunately, few contributions have been made to address inadequacies of the conventional technologies and effect of integration and threats information sharing in term of their dependency upon finite system resources to run ever growing threat signature database on the users' networks.

This growth occurs as a result of new threats signatures which are shared and added to existing threat database each time there is occurrence of new threats in the cyberspace. This could decrease network and system performance as growing threat signature database consume more host-based storage, memory and processing cycles. Also, most organizations; government, private and individual computer networks today do not have the complete in-house expertise and requirements to address the current emerging threats. This is because these requirements are either expensive or not available in the right proportion, hence the reason why so many of these attacks succeed. For instance, some organizations and individual may not afford expensive and licensed conventional network protection software to protect their computer networks. They resort to unlicensed software which cannot protect their information or network infrastructure.

To address these challenges, a model which can guard, maintain and detect attacks on users' networks beyond their capability is therefore proposed. The model which is cloud based will shift threat detection computation to the cloud

instead of depending upon finite system resources like processors, memory, storage and bandwidth of the users' networks to run the ever growing threat signature database. The proposed model will eliminate the time needed to create and distribute threats signatures each time new threats are detected. The proposed model has its theoretic framework rooted in intrusion detection (IDS) and cloud technology (Gul and Hussain. 2011).

There are quite a good number of works regarding the state- of- the- art network security and cyberspace protection. Wamala, 2011 proposed a cybersecurity reference model which nations could adopt and implement to suit their security requirements. The works of Govindarajan and Chandrasekaran, 2010; Tavalae *et al.,* 2009; Shanmugavadivu and Nagarajan, 2012 were based design and implementation of intrusion detection models as well as improvement on their performances in terms of accuracy and reliability to safeguard computer networks. Most existing works and solutions did not address the effect of users' networks performance degradation as a result of constant threat signatures information sharing inform of patches or software updates. Furthermore, most organizations do not have complete in-house expertise and requirements to address the current emerging threats, hence this study.

METHODOLOGY

The detailed description of the cloud based model for cyber threats management (CBCM) was done using Unified Modeling Language (UML). CBCM has two components: The user networks (UNG) and the cloud Server (CNTIS). The UNG is a secured independent server which is integrated into the users' networks using the proposed framework based on the requirements to protect their networks.

The main function of the UNG to filter all incoming packets for threats, if the packets are clean, they will be allowed into the network otherwise the UNG will quickly query the CNTIS to investigate the suspected malicious packet. The UNG which is anomaly based Intrusion Detection System (IDS) in nature was designed and implemented using brute force algorithm. With this design UNG is capable of detecting normal packets pattern only, any pattern deviating from known normal packet patterns are rejected and forwarded to CNTIS for investigation. Communication between UNG and CNTIS takes place via secured virtual private network (VPN) in an encrypted form so that hackers do not have access to the exchanged packets.

The CNTIS is made up of Cloud Guard module (CG) which has the same design and implementation as UNG and Threats Analysis Engine (TAE) module. The TAE was designed as a signature based IDS using ensemble of ANN.TAE is at the server side and if designed as signature based IDS, it will enable it to classify known threats and easily identify new threats which will be investigated and added to the threat database as they occur. Other components of CNTIS include; Threat Information Disseminator and Updater (TIDU) and Threats Signature Database (TSD).The different components of the CBCM were simulated in MATLAB 7.0 using NSL-KDD Dataset as evaluation data. NSL-KDD dataset is the refined version of KDD Cup'99. It has all the attributes and features of KDD Cup'99 dataset as well as training and test sets.

The sample version of the dataset has 65,535 connection records for training and 65,535 for testing dataset. 60% of the training and test sets were used for the experiment. The performance of the proposed model was carried out using two approaches: Remote Method Invocation (RMI) and Mobile Agent (MA). The performances of the two approaches were compared using response time of CBCM to detect threats, fault tolerant of the system and Bandwidth consumption metrics. The prototype implementation of the model was carried out using C# programming language. The conceptual view of the proposed model is shown in Figure 1, while Figure 2 is the system architecture.

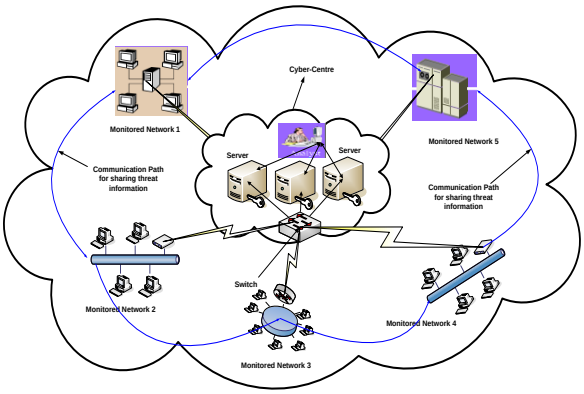

Figure 1: Conceptual View of the Proposed Model.

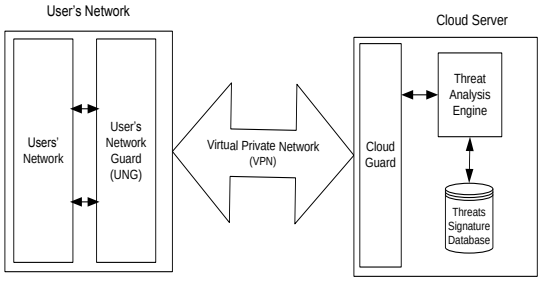

Figure 2: Proposed System Architecture.

MODEL DESIGN

For the purpose of implementation, the model proposed in this study was designed using Unified Modelling Language (UML). Among the UML diagrams used to design the model are; the use case diagram, class diagram and sequence diagram. The use case diagram of the CBCM is shown in Figure 3.The use case was used to describe the behavior of the model from the user's point of view. It showed the entities or objects involved in the system and different operations that the entities can perform. UNG, CNTIS and network security experts are the entities of the models. The UNG can perform operations like threats detection and new threat query. The CNTIS can perform novel threats analysis, updating threat database, respond to queries sent by UNG and dissemination of threat information. The experts can perform novel threat investigation.

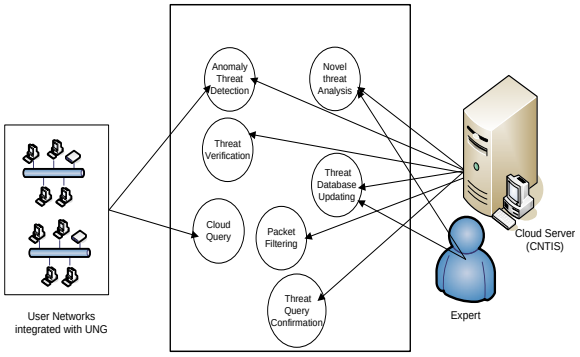

Figure 3: Use Case Diagram of the Proposed Model.

Figure 4 is the sequence diagram of the model, it shows how objects in the model interact over time. From Figure 4, when a packet arrives at user network, it passes through the UNG which will only identify normal packet. Any packet pattern that deviates from normal will be classified as malicious according to pattern of threat records in NSL-KDD.

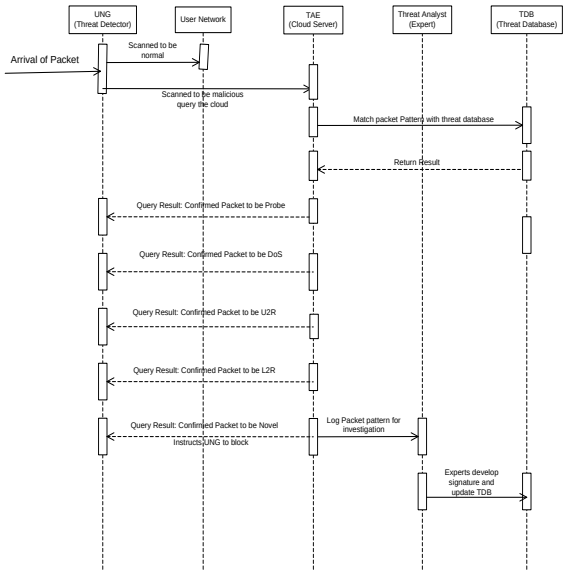

Figure 4: UML Sequence Diagram of the Proposed Model.

Packets classified as malicious are queried and sent to the CNTIS for analysis and investigation. The result of analysis and investigation component will cause CNTIS to respond on real-time to UNGs in the users' networks.

Figure 5 is the UML class diagram. This diagram shows the member variables, and functions of a class. The class dependencies and relationships are indicated in the diagram. For instance, the classes distributedNet and netServer are dependent. The class distributedNet report threats to the class netServer. The netServer in turn provides threat information and confirmation to UNGs that sent query. However, users' networks are updated if the packet queried is normal. A new threat discovered is investigated and users' networks are pre-informed of the pending threat prior to investigation.

The Design of UNG Module

The UNG is a component of the CBCM installed to guard users' networks and query the cloud server when anomalies are detected on the users' networks. The module was designed and implemented using Brute Force Pattern Matching Algorithm.

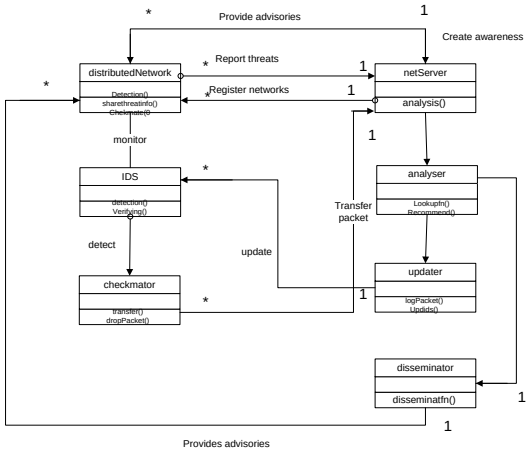

Figure 5: Class Diagram of the Proposed Model.

A brute force algorithm for string matching problem has two inputs to be considered: pattern (a string of m characters to search for), and text (a long string of n characters to search in). The algorithm starts with aligning the pattern at the beginning of the text. Then each character of the pattern is compared to the corresponding character, moving from left to right, until all characters are found to match, or a mismatch is detected.

While the pattern is not found and the text is not yet exhausted, the pattern is realigned to one position in the right and compared with the corresponding character, moving from left to right. In this study, "the string of *m* character to search for" is the packets coming into the users' networks which UNG screens for maliciousness. UNG is designed as anomalous intrusion detection. In this case it detects any deviation from normal packet patterns and such deviations are usually sent as query to the CNTIS for confirmation and investigation. The "long string of n characters to search in" is the normal packet patterns stored as in-built database in UNG. The in-built database in UNG contains the entire normal packet patterns extracted from NSL-KDD dataset used in this study. The brute force algorithm used in this study is presented as follows:

Algorithm BruteForceStringMatch
(T[0..n-1], P[0..m-1])
// Implement brute-force string matching
// Input: Text array T and pattern array P
//output: The index where P is found or -1
For I = 0 to n-m do
J= 0
While j < m and P[j] = T[i +j] do
j = j + 1
if j = m then return i
return -1

Design of Threat Analysis Engine (TAE)

TAE is a component of the CNTIS that performs analysis and initial investigation of packets suspected to be malicious and queried by UNG. The TAE takes as input, the malicious packet pattern and classify it according to the categories of attack classes in NSL-KDD dataset. TAE was designed as signature based Intrusion Detection System (IDS) using Artificial Neural Network (ANN) approach. The detail design mathematical description and performance evaluation of the threat classification model adopted for TAE is presented in Ogundoyin *et al.,* 2013.

The mathematical description of the Multi-layer perceptron (MLP) type of ANN model used for its design is stated as follows: The data attributes described in NSL-KDD dataset were used as input to the MLP model.

A = ' duration'
B = 'src_bytes'
AL = 'dst_host_srv_rerror_rate output'
$A_{out(1)}, A_{out(2)}, A_{out(3)}, \ldots\ldots\ldots\ldots\ldots\ldots, A_{out(k)}$

where, $A_{out(1)} - A_{out(k)}$ are output defined for the MLP model for TAE and k is the number of output specified.

To commence analysis and detection functions, the input data from NSL-KDD must be normalized. This is to remove discrepancies that could result from different data units and data types. The equation in (1) is the standard normalization formula.

$$Y_{norm} = \frac{Y_{ij} - Y_{min}}{Y_{max} - Y_{min}} \quad \textbf{(1)}$$

where Y_{norm} = variable containing normalized input data that is, A, B , C …… Ak, Al.

The normalized variables values will fall in range of 0 and 1

Y_{ij} = the data being normalized

Y_{min} = the minimum value of the data being normalized

Y_{max} = the maximum value of the data being Normalized

$$P_{input} = [A_{ij}\ B_{ij}\ C_{ij} \dots\dots\dots\dots\dots\dots\dots AK_{ij}\ AL_{ij}] \quad \textbf{(2)}$$

A_{ij}, B_{ij}, C_{ij} ……Ak$_{ij}$, Al$_{ij}$ are the elements of matrix P_{input}

where i , $1 \leq i \leq n$, n is the number of records in the data set

j, $1 \leq j \leq m$, m is the number of attributes contained in the data record used in the model

P_{input} = the variable name holding the input dataset for model training

D_{test} = the variable name holding the dataset for model testing

$$T_{target} = [A_{out(1,1)}\ A_{out(1,2)}\ A_{out(1,3)}\ A_{out(1,4)}\ A_{out(1,5)}\ A_{out(1,6)}] \quad \textbf{(3)}$$

$T_{target} = [A_{out(ij)}]'$ where i, $1 \leq i \leq k$ k = number of output defined for the model, and n, $1 \leq i \leq n$ where n is the number of records in the dataset

T_{target} is the variable name holding the target dataset.

$$X_1 = W_{11}A_{11} + W_{21}B_{12} + W_{31}C_{13} + \cdots + W_{381}AK_{138} + W_{391}AL_{139} \quad \textbf{(4)}$$

$$X_2 = W_{12}A_{11} + W_{22}B_{12} + W_{32}C_{13} + \cdots + W_{382}AK_{138} + W_{392}AL_{139} \quad \textbf{(5)}$$

$$X_3 = W_{13}A_{11} + W_{23}B_{12} + W_{33}C_{13} + \cdots + W_{383}AK_{138} + W_{393}AL_{139} \quad \textbf{(6)}$$

$$X_r = W_{1r}A_{ij} + W_{2r}B_{ij+1} + W_{3r}C_{ij+2} + \cdots + W_{38r}AK_{ij+38} + W_{39r}AL_{ij+39} \quad \textbf{(7)}$$

where i = 1, j = 1 and r is the number of neurons in the rth layer of the ANN

Therefore $X_1, X_2, X_3, \dots\dots\dots\dots\dots, X_r$ are passed into the transfer function chosen for the detection model

$$f(X_r) = \frac{2}{\{1+\exp(-X_r)\}^{-1}} \quad \textbf{(8)}$$

where r starts from 1 and is the number of neurons in the hidden layer

$$Y_1, Y_2, Y_3, \dots\dots\dots\dots\dots Y_r \quad \textbf{(9)}$$

then $Y_1, Y_2, Y_3, \dots\dots\dots\dots\dots Y_r$ are the outputs from the neurons in the hidden layer.

These outputs from the neurons in the hidden layer will serve as inputs to the neurons in the output layer of the ANN model

$$X_r^2 = Y_1W_1^2 + Y_2W_2^2 + Y_3W_3^2 + \cdots + Y_rW_r^2 \quad \textbf{(10)}$$

This equation can be rewritten as:

$$X_r^2 = \sum_{r=1}^{k}(W_r^2\ Y_r) \quad \textbf{(11)}$$

The sum of product of input and the connection weight at the hidden layer is also passed into the transfer function of the output neurons as in equation:

$$A_r^c = f(X_r^2) = \frac{2}{\{1+\exp(-X_r^2)\}^{-1}} \quad \textbf{(12)}$$

where r, $1 \leq r \leq k$

K is the number of neurons in the output layer which represent the output.

$A_r^C = is\ the\ computed\ attack\ pattern$

Therefore we can have:

$A_1^c, A_2^c, A_3^c, \dots\dots\dots\dots\dots, A_r^c$

The error E between the observed and the estimated attack pattern is computed as:

$$E = {}^1/_2 \sum_{i=1}^{n}[A_{out(i)} - A_{r(i)}^c]^2 \quad \textbf{(13)}$$

where n = number of records in the training dataset:

$$W_{(ij)n+1} = W_{(ij)n} + \Delta W_{(ij)n} + \beta\,[W_{(ij)n} - W_{(ij)n-1}] \quad \textbf{(14)}$$

where β is the momentum term.

The change in weight (ΔW) in the direction of negative gradient is given by:

$$\Delta W_{(ij)} = -\alpha\,\frac{\delta E}{\delta E_{(ij)}} \quad \textbf{(15)}$$

where α is the learning rate such that $0 \leq \alpha \leq 1$ and govern the rate of change of weight for the model.

MODEL PERFORMANCE EVALUATION

In order to evaluate the performance of the proposed CBCM, two communication approaches were considered for its simulation in order to determine a better approach for its implementation. This was considered since the proposed CBCM is made up of user network (UNG) and cloud server (CNTIS) components which exchange information inform of client-server system.

The two approaches considered are Agent (MA) and conventional Remote Method Invocation (RMI) approaches. Mathematical models were formulated based on the metrics: response time of the model to report or query suspected malicious packet, bandwidth of the model and fault tolerance of the proposed model in the face of failure or severe attack.

Two detection scenarios could occur in the experiment: The first is known threat detection in which UNG queries the CNTIS and appropriate response is immediately sent to the UNG that sent the query. The second detection is the novel threat detection in which CNTIS performs initial investigation, logs the threat for investigation by security experts and also sends notification to other users' networks about the pending malicious threat being investigated. The later scenario was considered in this study. The performance metrics model architecture for both MA and RMI are shown in Figure 6 and Figure 7. Table 1 shows the definitions of simulation settings.

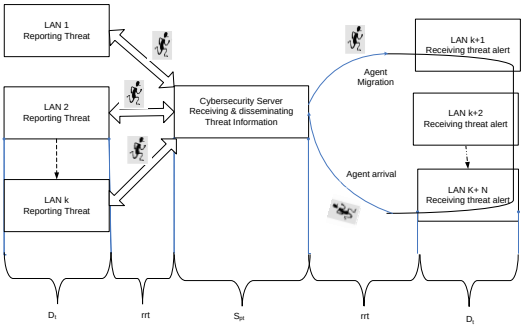

Figure 6: MA Modeling Architecture Components for the Threats Information Sharing Sub-Model.

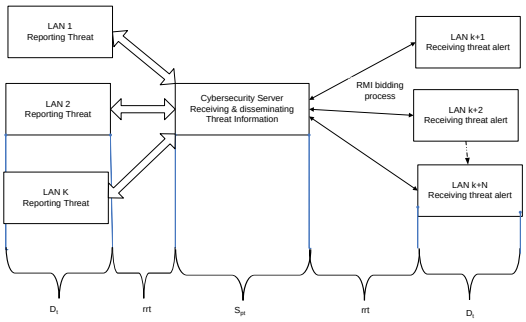

Figure 7: RMI Modeling Architecture Components for the Threats Information Sharing Sub-Model.

Table 1: Model Architecture Parameter Definition and Simulation Setting.

S/N	Parameters	Value
1.	Packet Size (PKt_s)	47936.56 byte
2.	Network Minimum Bandwidth (N_b)	2Mb
3.	Network Protocol	TCP
4.	TCP window size (TCP_{ws})	65536 bytes
5.	Server Processing time S_{pt}	Negligible
6.	Threat detection time D_t	0.00143s
7.	Round Trip Latency (rtt)	300ms
8.	Agent Code Size (X_{as})	47936.56 byte
9.	Network Size (N_s)	100

Response Time

Response time in this study is defined as the time interval between the time the threat detection module (UNG) of the user's network detects a malicious packet and queries the CNTIS (cloud server) to the time CNTIS sends the result to the UNG that sent the query and also inform other users' networks. The system is to provide a real-time identification of known threats and prediction of novel threats for analysis by experts at the CNTIS. The response time of the proposed CBCM when designed and simulated using MA and the RMI approaches are defined as follows:

(a) MA Model

From Figure 6 the mathematical model that describes the response time of CBCM using MA model is described:

D_t = Threat detection time of the attacked network
S_{pt} = Sever processing time

DES_{RMI} = The time used by server to inform other networks of newly detected threats:

Ns = [N1, N2, N3,……. Ns] **(16)**

Where s is number of networks in the cyberspace
N_k = Number of network that detect new attack.
Let,

$$R_{Agent} = N_k * (D_t + rrt) + S_{pt} + DES_{Agent} \quad \textbf{(17)}$$

R_{Agent} = MA response time
Where DES_{Agent} = MA dissemination time.

$$DES_{Agent} = \sum_{i=1}^{N_k} (rrt)_i \quad \textbf{(18)}$$

Putting Equation (18) in (17)

$$R_{Agent} = N_k * (D_t + rrt) + S_{pt} + \sum_{i=1}^{N_k} (rrt)_i \quad \textbf{(19)}$$

(b) RMI Model

From Figure 7 the mathematical model that describes this process is as follows:

DES_{RMI} = The time used by server to inform other networks of newly detected threats **(20)**

R_{RMI} = the time interval between when networks detect new threats, report the threats to the central server and the central server disseminate the threats information to other networks (LANs) using the proposed platform.

Therefore:

$$R_{RMI} = N_k * (D_t + rrt) + S_{pt} + DES_{RMI} \quad \textbf{(21)}$$

where DES_{RMI} = RMI dissemination time,
rrt = Round trip latency.

$$DES_{RMI} = \sum_{i=1}^{N_k} \sum_{i=1}^{N_s} (rrt)_i \quad \textbf{(22)}$$

Inserting Equation (22) in (21):

$$R_{RMI} = N_k * (D_t + rrt) + S_{pt} + \sum_{i=1}^{N_k} \sum_{i=1}^{N_s} (rrt)_i \quad \textbf{(23)}$$

Bandwidth Usage

The bandwidth consumption rate of the proposed model when implemented as MA and RMI are as follows:

(a) MA Model

In the MA model scenario, if the threat detected is novel, the agent arises from the threat detecting UNG and migrates to the cloud server, CNTIS. The CNTIS agent will then obtain the threat information and move round all the user' networks using UNG in the cyberspace in a single batch to inform them of the imminent threat information and return back to the CNTIS where it initially migrated. During visitation time, the message size is assumed to be size of average packet calculated from the packet size attribute of NSL-KDD dataset used in this study (that is, Y = 47936.56 byte). Therefore the agent migrates to the nodes with its code size X_{ag} and message size Y and return only with its code size. Therefore, the size for the complete journey is $2X_{ag}$ and number of LANs in the network is N. Hence the total bandwidth usage for the MA model is given by:

$$\beta_{Agent} = B_{cAg} + B_{sAg} \quad \textbf{(24)}$$

B_{cAg} = Bandwidth used by client agent
B_{sAg} = Bandwidth used by server agent

$$B_{cAg} = (X_{ag} + Y) + X_{ag} \quad \textbf{(25)}$$

where,
X_{ag} = Agent code size
Y = Packet size
if N_k network detect threats

$$B_{cAg} = ((X_{ag} + Y) + X_{ag}) * N_k \quad \textbf{(26)}$$

Equation (26) can be rewritten as:

$$B_{cAg} = (2 * X_{ag} + Y) * N_k \quad \textbf{(27)}$$

Bandwidth usage from the server side:

$$B_{sAg} = \sum_{i=1}^{N_k} (2 * X_{ag} + Y)_i \quad \textbf{(28)}$$

Therefore the total bandwidth for Agent model Putting Equations (27) and (28) in (24):

$$\beta_{Agent} = (2 * X_{ag} + Y)N_k + \sum_{i=1}^{N_k}(2 * X_{ag} + Y)_i \quad \textbf{(29)}$$

(b) RMI model

Let the message size from the user network, i.e. UNG be denoted by y and the size of acknowledgement from the server be z in bytes, for a single UNG in a LAN, the total bandwidth usage in bytes is given by

$$\beta_{RMI} = B_{client} + B_{server} \quad \textbf{(30)}$$

where,

β_{RMI} = Bandwidth consumption of RMI approach

B_{client} = Bandwidth used by client (UNG) to report threat to central server

B_{server} = Bandwidth used by CNTIS (server) to inform other user network,

Let message size from user network be Y and acknowledgement from server be Z:

$$B_{client} = (Y + Z) \quad \textbf{(31)}$$

if N_k client networks detect threat:

$$B_{client} = (Y+Z) \times N_k \quad \textbf{(32)}$$

Assume that Y = Z

Assume that the message size Y (bytes) is the same with acknowledgement Z (bytes) from the server i.e. Y = Z and it should be noted that a number of links must be established between the two communicating nodes before computation is completed then:

$$B_{client} = 2*Y \times N_k \quad \textbf{(33)}$$

From server side that is, B_{server}

Since the client networks will report to server whenever threats are detected, the server will in turn inform other networks and also consume bandwidth:

$$B_{server} = \sum_{i=1}^{N_k}\sum_{i=1}^{N_s}(Y + Z)_i \quad \textbf{(34)}$$

Therefore, the total bandwidth for RMI model in a particular time is:

$$\beta_{RMI} = 2 * YxN_k + \sum_{i=1}^{N_k}\sum_{i=1}^{N_s}(2 * Y)_i \quad \textbf{(35)}$$

Fault Tolerance

Fault is a measure of robustness or adaptability of a system to breakdown (Aderounmu, 2001). It is an ability of a system to respond to an unexpected hardware of software failure. In the case of this study, it is the ability of the system to respond to severe attack. To carry out the fault tolerance of the system, the proposed model was simulated as RMI and MA models. Probability of 0.1 failures was assumed (i.e. for every 10 nodes along path of communication, there is likelihood of fault or attack occurring in one (1) out of 10.

(a) **RMI model**

In the face of fault or network failure as a result of attack in RMI model, the model will not be able to scale. The RMI model will have to wait until the links are up again to continue operation because there is no way for the system to adjust dynamically to fault or its link failure. For instance, if a router along the path of a reporting UNG attached to a LAN fails, the query from UNG or reported suspected malicious threat information may not get to the CNTIS until the router or the failed node comes up again. That is to say that the system will experience delay equal to the node recovery time *nRt.* This will be added to the normal response time of the system when there is no failure. When failure of the node occurs, fault tolerance can be modelled as follows:

$$R_{RMI} = N_k x(D_t + rrt) + S_{pt} + \sum_{i=1}^{N_k}\sum_{i=1}^{N_s}(rrt)_i + nRt \quad \textbf{(36)}$$

(b) **MA Model**

In the face of failure or faulty links, the mobile agent can proactively determine alternative route in order to connect to the next node so that the destination could be reached. This can be dynamically achieved using shortest path Floyd algorithm. This will make the response time to be unchanged compared to when there is no fault or attack.

$$R_{Agent} = N_k * (D_t + rrt) + S_{pt} + \sum_{i=1}^{N_k} (rrt)_i \quad \textbf{(37)}$$

RESULTS AND DISCUSSION

The mathematical models formulated for the performance metrics in this study were simulated, the results are presented as follow:

Response Time

The response time results in both RMI and MA models were compared against the number of UNG reporting suspected malicious threats in a particular time to the CNTIS. The network used as a test bed in this simulation can only transmit 2Mb/s. Figure 8 is the simulation result of the model response time, when modelled as RMI and MA. The results showed that MA model performed better than the RMI model. The better performance of MA in term of response time is as a result of lack of multiple connections even before exchange of actual data which characterized RMI model approach. For instance in Figure 8 when the number of UNGs reporting threats is 45, the response time in RMI approach is 4050.06s and the corresponding MA approach generated response time of 2727.06s.

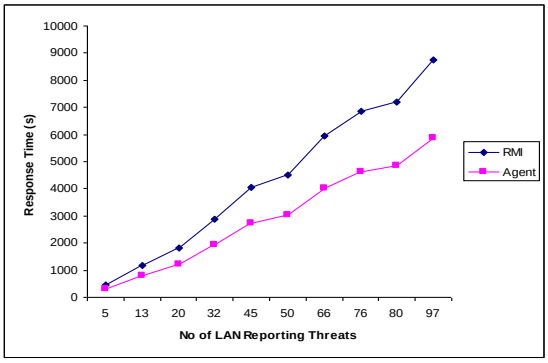

Figure 8: Response Time (RMI and MA).

Bandwidth

The bandwidth usage was measured by running the developed simulation program several times. Random number generator was used to simulate and generate the number of UNG reporting threats at a particular time for both RMI and the MA approaches.

Figure 9 shows the simulation results obtained for bandwidth usage versus the number of UNGs generating threat reports in the cyberspace for both RMI and MA approaches. For instance, RMI consumed bandwidth (Kb/sec) of 1551360, 40335440, 6205440 and 9928700 while MA used bandwidth (kb/sec) of 61440, 159744, 245760 and 393216 when the number of UNGs generating threat reports were 5, 13, 20 and 32 for both the two approaches (RMI and MA). The results obtained obviously showed and confirmed that MA approach performed better than RMI.

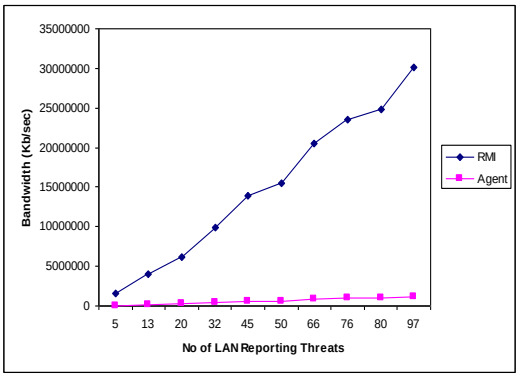

Figure 9: Bandwidth Consumption (RMI and Agent Approaches).

Fault Tolerance

Fault is unavoidable in any system, but occurrence of fault has to be kept very low in certain critical system like the CBCM developed in this research. To measure the fault tolerance of the developed model, the response time of the model in both RMI and MA paradigms were measured during fault occurrence. Figure 10 shows the result.

The better performance of MA paradigm can be connected to more local invocation made as against the RMI which relies so much on remote (network) connection throughout any service, therefore making it more susceptible to network failure. The increased response time in RMI is also due to the network or node recovery time used in the simulation.

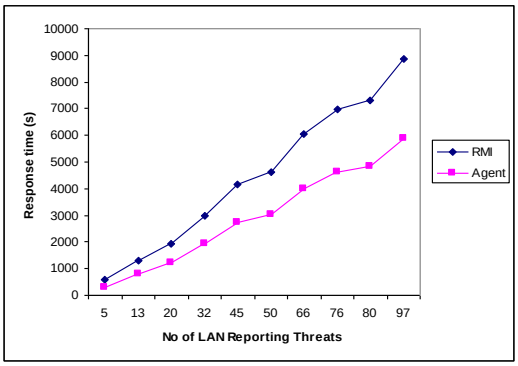

Figure 10: Fault Tolerance of the Model (RMI vs. MA Approaches).

IMPLEMENTATION

The prototype of the cloud based model for threats management in this study was implemented based on MA architecture approach using C# programming language. This is as a result of better performance of MA approach during simulation experiment. The prototype implementation was demonstrated on a private LAN with five computer systems. One of the systems was used as server which represents CNTIS and the remaining four systems on the LAN were assumed to be users' networks on which UNG was installed.

The UNG which is a threats detector (IDS) detected threats challenges sent to it. The screen shot captured during test running of the prototype software for the network threat management were presented: Figure 11 is the screenshot of the UNG module interface which is the interface through which user can interact with the software.

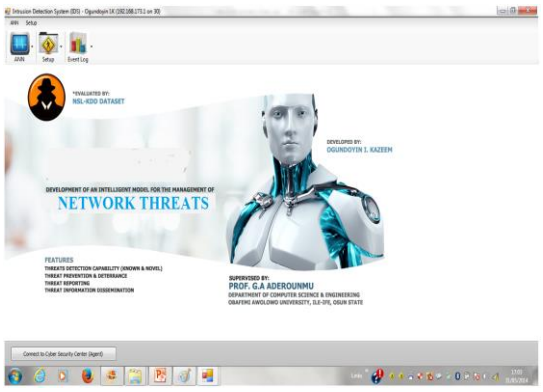

Figure 11: Screenshot of Client Module.

Figure 12 is a screenshot of CNTIS module interface which receives suspected malicious novel threats for investigation. Through this module, security alert and threat information generated by the CNTIS are sent to all UNG installed on users' networks can be viewed.

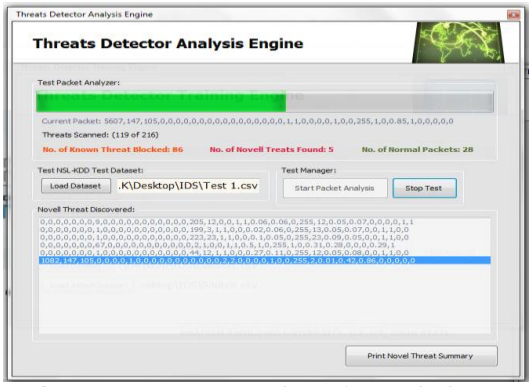

Figure 12: A Screenshot of Novel Threat Investigation Interface.

CONCLUSION

In this paper a model was proposed with emphasis on shifting threat detection computation to the cloud where there is enough computational resources and expertise to mitigate threats. With this method the effect of constant and new threat signatures information sharing known to consume resources and degrade users' network performance has been addressed.

The developed model has the potential to greatly enhance cyberspace protection and could be used for national cyberspace monitoring. Issues of trust and reputation as related to security of information to be shared in the cloud will be investigated in the future work.

REFERENCES

1. Aderounmu, G.A. 2001. "Development of an Intelligent Mobile Agent for Computer Network Management". Ph.D. Thesis, Department of Computer Science and Engineering, Obafemi Awolowo University: Ile-Ife, Nigeria.

2. Adnan, N. and P.H. Michael. 2014. "An Intrusion Detection & Adaptive Response Mechanism for MANETs". *Ad Hoc Networks*, 13(2014):368-380.

3. Bin, L. and X. Jingbo. 2014. “A Novel Intrusion Detection System Based on Feature Generation with Visualization Strategy”. *Expert Systems with Application*. 41(9):4139–4147.

4. Bul'ajoul, A., W. James and M. Pannu. 2015. “Improving Network Intrusion Detection System Performance through Quality of Service Configuration and Parallel Technology”. *Journal of Computer and System Sciences*. 81(6):943–957.

5. Gisung, K., L. Seungmin, and L. Sehun. 2014. “A Novel Hybrid Intrusion Detection Method Integrating Anomaly Detection with Misuse Detection”. *Expert system with Application.* 41(4):1690–1700.

6. Govindarajan, M. and R.M. Chandrasekaran. 2010. “Intrusion Detection using Neural Based Hybrid Classification Methods”. *Elsevier, Computer Networks Journal*. 55:1662–1671.

7. Gul, I. and M. Hussain. 2011. “Distributed Cloud Intrusion Detection Model”. *International Journal of Advanced Science and Technology*. 34 1-82.

8. Jama, l H. and L. Samuel. 2016. “Feature Analysis, Evaluation and Comparisons of Classification Algorithms Based on Noisy Intrusion Dataset”. 2nd International Conference on Intelligent Computing, Communication & Convergence, ICCC 2016. 24-25 January 2016. Bhubaneswar: Odisha, India. 92: 188–198.

9. Kumar, N., J.P. Singh, R.S. Bali, S. Mistra, and S. Ullah. 2015. “An Intelligent Clustering Scheme for Distributed Intrusion Detection in Vehicular Cloud Computing”. *Journal Cluster Computing*. 18(3): 1263 – 1283.

10. Mafra P.M., J.S. Fraga, and A.O. Santin. 2014. “Algorithms for a Distributed IDS in MANETs” *Journal of Computer and System Sciences*. 80(3): 554-570.

11. Nabil E.K., H. Karim, and E.Z. Nahla. 2012. “A Mobile Agent and Artificial Neural Networks for Intrusion Detection”. *Journal of Software*. 7(1):156–160.

12. Ogundoyin, I.K., E.A. Olajubu, S.A. Akinboro, and G.A. Aderounmu. 2013. “Development of an Improved Intrusion Detection System for cybersecurity Threats Management”. ICEE/ICIT-2013 Proceeding Cape Peninsula University of Technology: Cape Town, South Africa. 40 - 48.

13. Shanmugavadivu. R. and N. Nagarajan. 2012. “Learning of Intrusion Detector in Conceptual Approach of Fuzzy Towards Intrusion Methodology”. *International Journal of Advanced Research in Computer Science and Software Engineering*. 2(5):246–250.

14. Tavallaee, M., E. Bagheri, W. Lu, and A.A. Ghorbani. 2009. “A Detailed Analysis of the KDD CUP 99 Dataset”. *Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defence Application (CISDA 2009)*.

15. Vance, A. 2014. “Flow Based Analysis of Advanced Persistent Threats”. First International Scientific-Practical Conference, Problems of Info communications. IEEE PIC S&T. Kharkiv, Ukraine. 173 – 176.

16. Wamala, F. 2011. *The ITU National Cybersecurity Strategy Guide*. International Telecommunication Union (ITU).

SUGGESTED CITATION

Ogundoyin, I.K., C.O. Akanbi, A.A. Adigun, and S.A. Akinboro. 2017. “A Cloud-Based Model for Computer Network Threats Management”. *Pacific Journal of Science and Technology*. 18(1):152-162.

Pacific Journal of Science and Technology