An Artificial Immune Networking Using Intelligent Agents

S. Chandrasekaran[1], C.Dinesh[2], AL.Murugappan[3]

ABSTRACT

The objective of the proposed work is to formally verify and implement an Artificial Immune Networking model. The work focuses on the protection against computer viruses which disrupt the normal usage of the network. The vulnerability of the network due to the malfunctions is detected in terms of faulty or malicious nodes. An Artificial Immune System or Bio-Inspired approach is adopted by making the network to automatically detect and tolerate the previously unseen normal behavior. In this system approach the dynamic description of the usual behavior of network is formally specified and clustering of decision making methods to minimize the time of detection. The theoretical analysis of such Immune Networking is also performed and the necessary network components are proposed to counter attack the intrusions. The Artificial Immune Networking and its timely response are modeled using Colored Petri Nets (CPN) at the illegal intrusions. The detection and recovery from the intrusion may be achieved by employing intelligent agents in a distributed environment. The performance of the system is determined by the correct activation of the lympho component at the earliest within the scope of intelligence given to individual agents.

Categories and Subject Descriptors
K.6.5 [**Security and Protection**]: Authentication, Invasive software (e.g., viruses, worms, Trojan horses)

General Terms
Security

Keywords
Artificial Immune Network, Colored Petri Nets, Lymphos, Intelligent Agents

S.Chandrasekaran is with Dr. M.G.R University, Chennai,, India Working as Professor and Dean of Computer Science Department(email: chandrasekaran_s@msn.com, mobile:+91 9841983789)
C. Dinesh is with Polytechnic University, Brooklyn, NY, (email: dchand01@students.poly.edu ,mobile:732 762 2220)
AL.Murugappan is with K.C.G College of Technology ,Chennai,India(email:murugappan.alagappan@gmail.com ,mobile: +91 9444586197)

1. INTRODUCTION

In the human oriented computational world, there has been a growing interest in the use of biology as a source of inspiration for solving computational problems. There are several computational techniques that look to biology for inspiration. Some common examples include networks, evolutionary algorithms, and Artificial Immune Systems or Immunological Computation [2]. The proposed work is often referred to as a strong member in Biologically Inspired Computing. The motivation of this work is primarily to extract useful mechanisms or metaphors from natural biological systems, in order to develop cost effective computational solutions to complex problems in a networking domain. [6], [9] proposed the use of immune system concepts for design of computer security systems and provided an elaborate description of some immune system principles applicable to security. The immune system also contains many useful information-processing abilities, including pattern recognition, learning, memory and inherent distributed parallel processing called as Artificial Immune Networking Systems (AINS).

Essentially, AINS are the use of immune system components and processes as to construct computational systems. Applications of AINS include such areas as mobile networking, business process prediction, fault diagnosis and tolerance computer security, scheduling, virus detection, and optimization. The work presents a general introduction to the field of artificial immunology, stressing the key areas that are currently used within the field of AINS. But the applications of these artificial immune networking will give the expected security and performance from the common internet users only when the networking is modeled in a more formal way. Hence the focus is to arrive at a computationally intensive and distributed model for the network component immunity using a powerful PetriNet modeling approach. An event triggered approach is used where the interaction with other systems in the network is asynchronous in nature. Based on the detection of any intruder or virus into the network, the response for the malicious attack is concurrent in its correction phase.

1.1 Immune Model – Acquired Immunity

The two ways that any biological system responds to a virus are suppression and tolerance. Immunity may be called as the degree of suppression and tolerance is the degree of acceptance. The tolerance of the computer network against viruses is the cross reactivity **to** agents and system security components added to the system. It is highly specific for each virus so that the network needs special detection and vaccination software components. But the immunity may be natural or acquired, say the system components and the external security activities respectively as shown in Table 1..

Table 1. Bio and Artificial Immune mapping

Biological Immune System	Artificial Immune System
Human Body	Computer Network
Organisms/ Organs	Nodes / Files
Antibodies	Mobile Agents
Antigens	Software Virus
Immunity, Suppression	Immunity, Tolerance
Neural Controller	Server
Learning	Training the agents
Immune memory	Look up Table
Training patterns	Virus Signatures
Receptors	Detectors
Bio Connectivity	Wireless/ Wired Link
Organ address	IP Address
Time of Attack	Time of Virus Detection
Cloning	Agent Replication
Recovery Time	Agent Life Time
Natural Immunity	Built –in Security
Acquired Immunity	Agent based Security
Natural Death	Dead PC

File infector viruses work by inserting their code into executable files, just as the biological virus works by inserting its DNA code into living cells [4]. When the virus or antigen enters the host computing node at a specific file, the host starts producing the antibodies in response to those interrupts. The antibodies produced with in the system software for security management modules and the resistance offered to any malicious code may be called as natural or self immune networking. If the system is introduced with some amount of antibodies or the network is induced to create antibodies for certain antigens or viruses, the mechanism may be called as acquired immunity. The point to be highlighted here is that the antigen what is going to be injected is a processed antigen like the dynamically created agent object. The negative effect of that induced agent is highly minimized through code optimization and scanning the same agent code multiple times. The processed antigen is a foreign body so that it will start producing antibody for it. Hence the network is immune to that particular virus only. As the network functions or expands over any period of time and if it is exposed to the same type of virus once again, it generates similar antibodies or agents, the agents would not allow the virus getting propagated from the affected node to another one. The natural immunity is any standard virus detecting mechanisms available only in the host system whereas the acquired immunity is developed not only by the agents roaming in the network but also by the way of replicating their own agents depending upon the nature of virus attack.

Some deadly viruses create noticeable amount of network performance degradation, sudden system reset, drivers malfunctioning and loss of files leading to Acquired Immune Deficiency Syndrome (AIDS) of a computer network. This type of virus enters the host and suppresses the immunity of the network so that other viruses can easily intrude into the network and thereafter make the network, not respond properly to the future injected antibodies. The entire network has to be finally shutdown loosing all data and connectivity. The system has to be reinstalled under a long maintenance period otherwise it has to be replaced. When the artificial immune network is attacked by an external antigen say a virus, there are two possibilities: tolerance or immunity; the correct selection is crucial and depends on many factors, like the nature, concentration of the antigen and the place of attack. It is assumed that low concentration of antigen normally induce tolerance; whereas medium concentration induce immunity. Of course, high concentration also induces tolerance, but the mechanism is not the same as for low concentration of the antigen in the network. When an antigen attacks a file, the artificial immune network decides which pathway should be taken either suppression or immunity and it has to remember the choice for a long time, even after the disappearance of the antigen as in the biological system. It is crucial that the infected file does not react against some of its other native files.

The agent server in the immune system produces agents (antibodies) directed against the infected file. The phenomenon of learning, i.e., the network has to learn whether to suppress or fight the incoming antigen, is massively parallel. Since the type and size of the all antigens are extremely large, the complexity of the learning process is cumbersome. With finite number of known antigens it is possible to train the network for early detection and correction of the antigens which are non- self, since the process is sequential. The degree of resistance is a function of the number of antigens entered, their size and also their severity along with the place of attack whereas the degree of tolerance is a function of the number of antibodies injected or created at that point of time in the network and their detection capability in terms of the number of training patterns.

This approach for the design and development of an Immune Network for virus detection is more robust due to the possibility of agent creation and cloning to decrease the total storage need of the virus signature list at every host. The autonomous, multilayered, and distributed features of this Biological Immune Systems suggest a distributed Mobile Agent System utilizing a diverse array of agent detectors [10].

2. ACQUIRED IMMUNITY PROBLEM FORMULATION

The acquired immunity in the distributed network has been illustrated with mathematical parameters representing the network's sensitivity and specificity. The immune network sensitivity may be calculated as the ratio of the product of the number of mobile agents created and the time taken to detect the malicious codes in a single node and report them to the central agent server to the time taken to detect the presence of all worms which attack the whole network at the same duration of time. The specificity represents the ratio of the number of specific infections or worms detected by an agent with or without replication to the total number of detectable worms in the given network.

Let us assume a distributed network with a number of computing nodes, say N. The Roaming Agent (RA) visits all nodes whose Internet Protocol (IP) addresses are stored in the IP Table. The Network Server (NS) is capable enough to create as many number of agents or copies of any agents depending upon the nature and place of attack. The number of agents created at any instance of time is represented by the set A. The list of known worms or virus signatures, say V are stored in the server.

$$\mathbf{N = \{n_1,n_2,n_3\ldots\ldots n_i\ldots\ldots. n_N\}}$$

$$\mathbf{A = \{a_1,a_2,a_3\ldots\ldots a_j,\ldots\ldots. a_M\}}$$

$$\mathbf{V = \{v_1,v_2,v_3\ldots\ldots\ldots v_k\ldots\ldots\ldots vp\}}$$

Let the RA starts roaming through the network at t_0. Let the time of status report by RA to Network Server be t_1 from the node n_i.

Let the time taken by the NS to dispatch the correct mobile agent a_j in the form of a java class file to the suspected node as per the IP-Address table be t_{cd}, the time for creation and dispatch. The traveling time of such a class file is taken to be t_t. The time taken to find a match for the substring is t_s, scanning time.

The compilation in the available Java Run Time Environment (JRE) in the affected node creates an antibody to scan the file for the presence of an antigen in the specified file path. The path is targeted towards any ***.exe*** or ***.zip*** or it may be any file type. The immunity of the network lies in the activation of the Lymphocytes (JRE) by the mobile agent created in the server. The activated antibody scans the specified file in its correct location for the presence of worm that is in the form of first 65536 bits or any length as per the developer's choice depending upon the severity of the attack, machine code in the case of any executable file. The scanning is a fast pattern matching problem looking for a substring – the worm string in the first 2 ^ 16 bits of the entire machine code. If the antibody detects a substring and verifies it with the known worm signature, it reports to the server through an agent but at this time, the return agent is infected and dies at the server after having entered a report in the log available at the server. Next the server takes a copy of the same agent class by duplicating the code and sent to next suspected address or creates a new agent class for another form of suspected attack. The total time needed to detect, create, dispatch and scan all the nodes against the worms may be calculated as,

$$T_{total} = N \times (t_0 + t_{cd} + t_t + t_s)$$

$$\text{The sensitivity of the AIN} = \frac{\text{Detection time for a worm at the } i_{th} \text{ node by } j_{th} \text{ agent for signature } v_p}{\text{Total time for detecting and Scanning all nodes against all signatures, } \mathbf{T_{total}}}$$

$$\text{The specificity of the AIN} = \frac{\text{No of infections detected at the } i_{th} \text{ node by } j_{th} \text{ agent for signature } v_p}{\text{Total worms detectable at all nodes against all signatures, V}}$$

3. FORMAL DESIGN USING COLORED PETRI NETS

Colored Petri Nets (CP-nets or CPN) is a graphical oriented language for design, specification, simulation and verification of systems. It is in particular well suited for systems in which communication, synchronization and resource sharing are important. Typical examples of application areas are communication protocols, distributed systems, automated production systems and work flow analysis. The development of CP-nets has been driven by the desire to develop a modeling language – at the same time theoretically well-founded and versatile enough to be used in practice for systems of the size and complexity we find in typical industrial projects. Petri nets provide the primitives for the description of the synchronization of concurrent processes, while programming languages provide the primitives for the definition of data types and the manipulation of data values.

The CPN modeling language combines Petri nets, which provide the foundation of the graphical notation and the semantical foundation for modeling concurrency, synchronization, communication in systems and programming languages. The module concept is hierarchical, allowing a module to have a number of sub modules and allowing a set of modules to be composed to form new modules. This enables the modeler to work both top-down and bottom-up when constructing CPN models. CPN models can be timed, meaning that the time taken by different events in the system can be modeled. This means that CP-nets can be used to investigate both logical and functional properties such as absence of deadlocks, and performance properties such as execution times and queue lengths [5].

A CPN model can be verified, to prove that it behaves as desired. The most straightforward verification method builds on model checking by means of state spaces, i.e., directed graphs with a node for each reachable system state and an arc for each possible transition from one system state to another. State spaces often become huge. Design/CPN also supports construction and analysis of state spaces, allowing the user to verify a large variety of different behavioral properties. The CPN language, i.e., the syntax and semantics can be used to characterize the behavior of a given CP-net, e.g., reachability, boundedness, home states, liveness and fairness. The formal analysis and design methods covering state spaces, place invariants, transition invariants and timed CP-nets are discussed in this work. The CPN Model of the Mobile agent based Artificial Immune Network is shown in Figure 1.

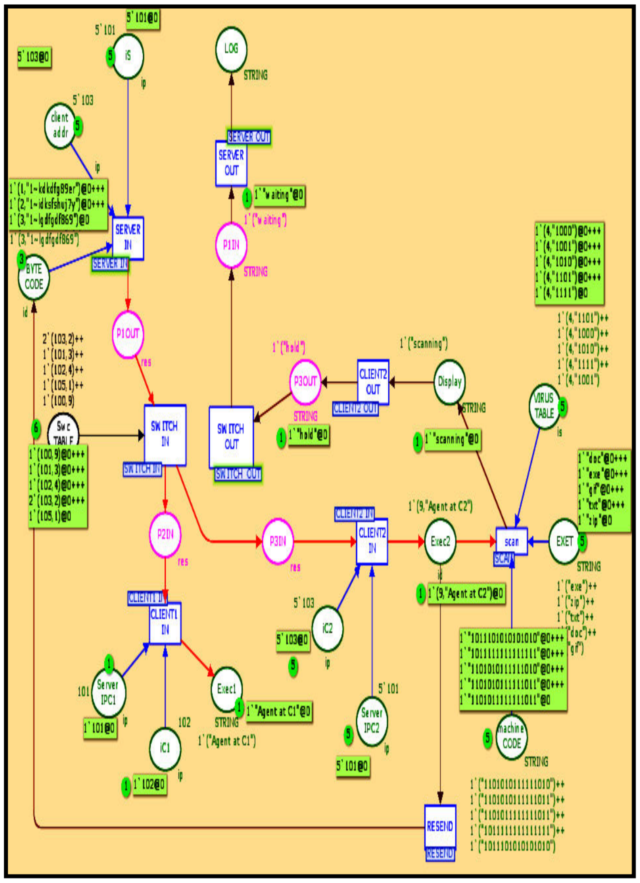


Figure 1. CPN Model of AINS

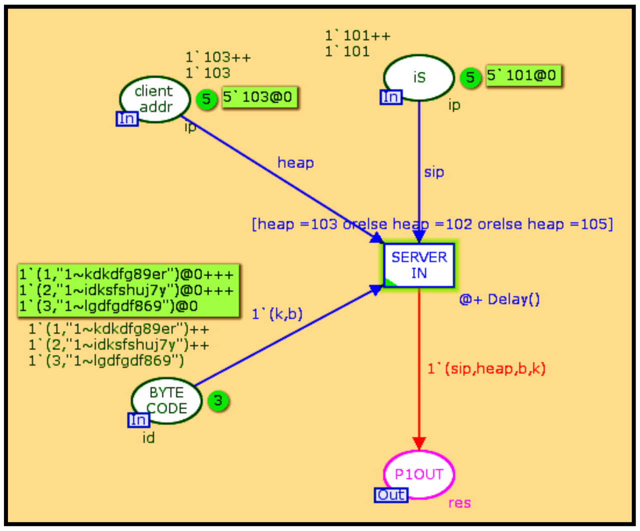


Figure 2. Mobile Agent Creation Scenario

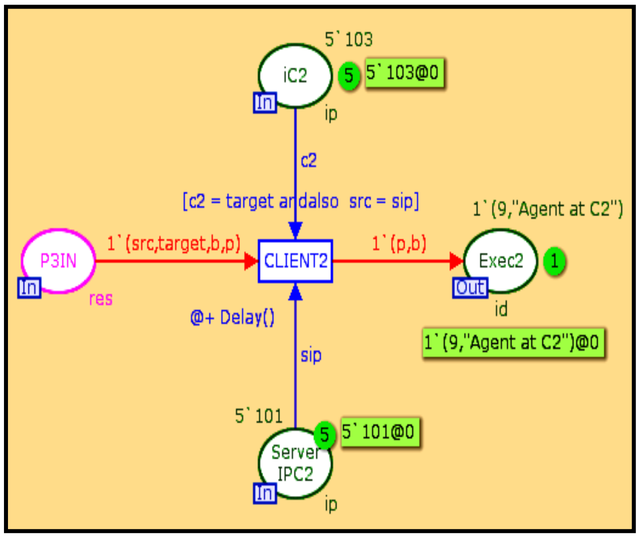


Figure 3. Host Verification Scenario

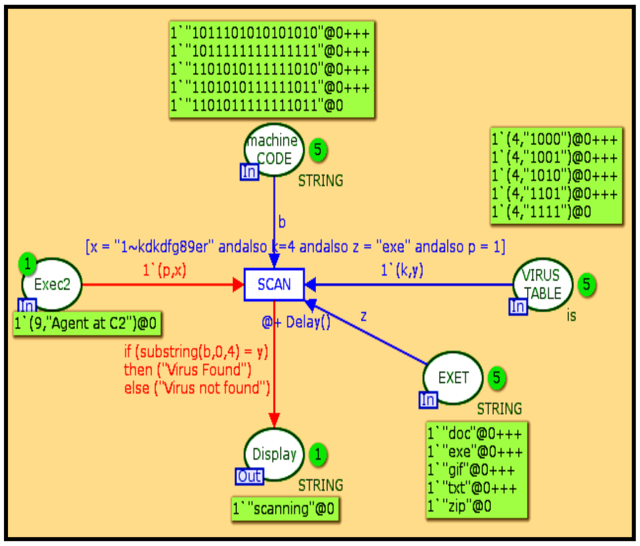


Figure 4. File Scanning Scenario

In the agent creation scenario, the agent packet is formed using the following tokens [BYTE_CODE, Client_addr, Server_addr] which is dispatched after specific time delay as shown in Figure 2. Timed protocols have been used to incorporate unknown delays that might be caused at every hardware. After agent authentication, the executable files are scanned for virus signatures stored at the server and the log file is updated as shown in Figure 3 and Figure 4. The basic software components that are needed in an artificial immune networking are roaming agent component for virus detection in an intelligent way and the scanning agent modules for actual virus scanning like lymphocytes. These lymphocytes like components are generated from the central server module. Different types of scanning agents as per the nature of attack, i.e., virus signatures are generated and the correct scanning agent is selected from the server in an intelligent way. The agents can be replicated if the number of similar attack is concurrent.

4. ARTIFICIAL IMMUNE NETWORKING COMPONENTS

The key portion of antigen that is recognized by the antibody is called *epitope*, which is the antigen determinant; *Paratope* is the portion of antibody that corresponds to a specific type of antigens. Once an antibody combines an antigen via their epitope, and paratope, the antibody starts to eliminate the antigen. Recent studies in immunology have clarified that each type of antibody also has its own antigenic determinant, called an *Idiotope* as shown in Figure 5. This means an antibody is recognized as an antigen by another antibody. Based on this fact, scientists proposed the concept of the *immune network*, or *idiotypic network*, which states that antibodies and lymphocytes are not isolated, but they are communicating with each other. Thus, the immune response eliminating foreign antigens is offered by the entire immune system (or, at least, more than one antibody) in a collective manner. In the proposed Artificial Immune Network, the infected portion of any file, say the first 65536 bits or the last 1024 bytes constitute a "worm" or an antigen is identified by the antigen specific "java class" – Software Agent.

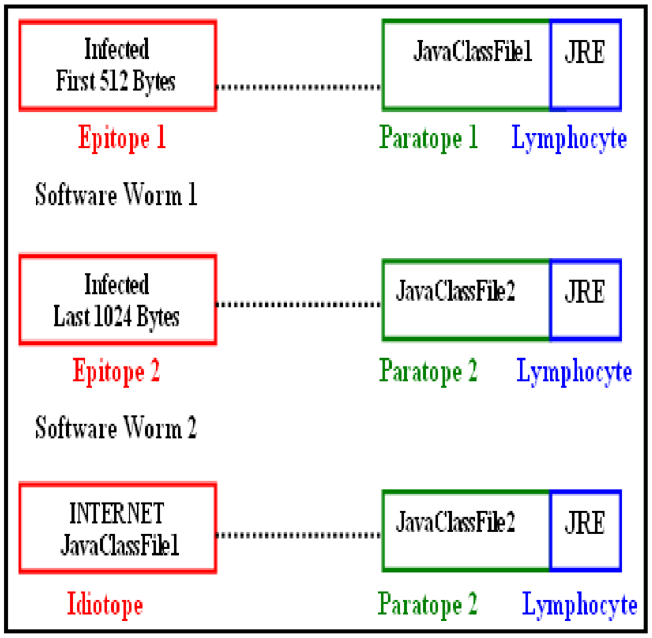


Figure 5. Artificial Immune Network Components

The architecture of immune network can be explained with the help of the following functional modules interconnected in a layered fashion as shown in Figure 6. The bottom most communication layer corresponds to the distributed transport and internet protocol stack for Wireless or LAN environment. The IP-addresses of the participating nodes are often visited and their links are checked by a roaming agent to make a First Information Report (FIR) regarding the suspicious node against the unwanted malicious code fragment. The FIR is prepared based on the Cyclic Redundancy Check (CRC) mechanism. The health condition or status of the node is manipulated by the server side component and a suitable agent is created that carries a java class file to suspected node. The creation and dispatch of the correct and needed class file could be achieved by training the network. The next layer corresponds to the actual scanning of the file in the specified location for the presence of worm in the first 65536 bits of the file that may be a .exe or .zip format. The upper layer is needed for the formal report on the post scanning process including the number of files scanned, the time of detection of worms if any and the reporting time of the scanning agent to the server after it returns back to the server.

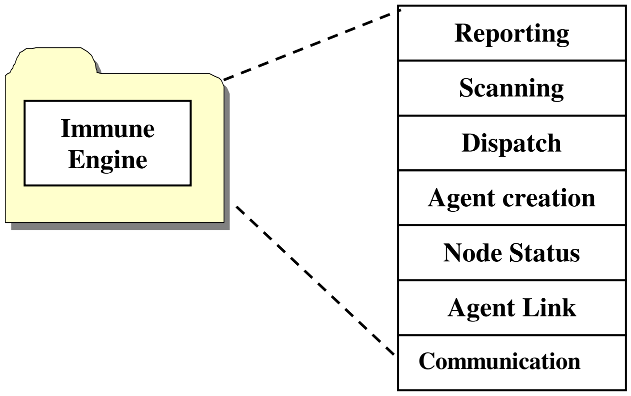


Figure 6. Layered Architecture

4.1 Virus Signatures and Detection

The Virus Signatures are unique bit strings or binary pattern, of a virus. It is like a fingerprint in that it can be used to detect and identify specific viruses. All the commercially available Anti-virus softwares use the virus signatures to scan for the presence of malicious code. In general the machine code of the virus program has to be compared for match with the code of file under scan. Typical computer viruses are a few thousand bytes in length. And given that there are several thousand PC viruses the amount of memory required just to contain all of the patterns would be several megabytes. It would be much easier for the virus authors as they have been given access to such virus databases. To avoid such practices instead of an exact match, the softwares use just a small piece of the virus code for scan. These short templates, called signatures, improves the performance of the scanner and also reveal nothing useful to virus authors. The algorithm as shown in the box that is used for virus pattern matching is simple. The machine code of the file under scan is pattern matched with the virus templates [7]. The tested positive files are deleted to avoid further damage to system softwares.

Pattern Matching Algorithm:

```
define V = {v1,v2,v3…vj}; // Set of Virus Signatures of length $l_e$

define F = {$f_1$,$f_2$,$f_3$…$f_p$} ;  // Set of executable files to be scanned
define M = {$m_1$,$m_2$,$m_3$…$m_k$} ;// Set of m. codes of files scan

   int   i, j, k, p  = 0 ;  //  Increment variable initialization
SCAN :     if $v_j$ =  SUBSTRING ($m_k$, i, $l_e$)
                then   i++ ;
          Loop till LENGTH ($m_k$);  DELETE ($f_p$) ;  EXIT
```

5. INTELLIGENT AGENTS FOR AINS

5.1 Scenario in Distributed Systems

The biological system is made up of many different types of cells that operate independently, yet in cooperation with each other in order to protect the body from foreign invasion. This highly parallel and distributed structure suggests that an integrated architecture can be viewed as a multiagent system (MAS), where separate functions are carried out by individual agents [3]. Mobile agent technology offers a new computing paradigm in which a program, in the form of a software agent, can suspend its execution on a host computer, transfer itself to another agent-enabled host on the network, and resume execution on the new host. According to definition, a multiagent system is a distributed computing system consisting of interacting agents that coordinate their actions inorder to complete competitive tasks [1]. To detect pathogens, the signature of machine code of the file is matched against signatures of potential viruses stored in an immune system database. The Bio Inspired Network model operates in two different modes; Normal Mode and Bio Mode. Firstly the Intelligent Roam Agent travels through the network, visits each and every node to detect changes in the CRC of windows executable files confined to a specific directory, say C:\WINDOWS\ where system files necessary for normal windows boot up resides. And at the end of every round trip it calculates the number of infected nodes. The Intelligent Roam Agent, depending upon that number, initiates one of the modes either virus tolerance or suppression.

In the Normal Mode, the master agent created migrates and scans executable files under the specified directory in all the hosts connected to the server within the domain. The signature database present only at the server is queried by the agent during its travel using Remote Method Invocation – Java Data Base Connectivity (RMI – JDBC). The results are updated in a log file created at the end of scanning process. If the log has more than the tolerable number of suspicions, then the agent terminates the host computer to prevent further damage.

5.2 Concurrency and Self – Replication of Mobile Agents

In the Bio mode of operation, the master agent performs self – replication to create its own multiple instances to solve concurrent attacks. The severity of the attack is calculated by the Intelligent Roaming Agent. In both modes, the first 2^16 bits are verified against 18 – bit virus signatures. The Cyclic Redundancy Check value of the executable files stored in the database is compared to detect the attack or corruption by viruses. The Agent calculates and checks its own CRC before execution, which will disable the virus from further multiplication. The proposed model has several characteristics. They are

- Captures missing code from local class loader – Self Healing
- Clones itself when concurrent attacks are detected – Self Replication
- Finds the path to the destination Autonomously – Self Adaptation

The traditional method of virus detection and correction requires the commercially available scanners to be present in all the host machines within the Local Area Network (LAN). The virus definition databases should be updated at regular intervals using the Internet to prevent attacks of newer viruses in the server. The proposed Bio Inspired Network model as shown in Figure 7, solves the specified constraints (Cost and Storage Capacity) faced by the traditional methods followed in the computer laboratories by maintaining a single virus signature database and scanner using Intelligent Agents. The model requires the agent server to be running at every host to receive and forward the incoming software agents. The RMI – JDBC is started at the server for the mobile agents to query the master database for signatures and CRCs.

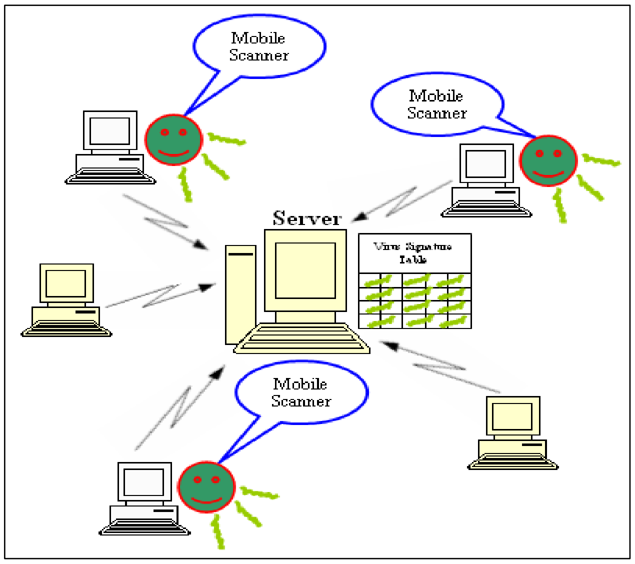


Figure 7. Proposed Bio Inspired Network model using Intelligent Agents

The agents are created using Kaariboga; a free implementation of a framework for mobile agents [8]. It is implemented in Java and thus it supports dynamic class loading and can be used on every computer supporting the Java platform. The server starts with default security manager and hence vulnerable under open networks. The Kaariboga project developed at the university of denmark lacked the implementation for code mobility and required the class file of the agent at all hosts. This feature is included to enhance its functionality, performance and to support code mobility.

Kaariboga uses TCP/IP for agent migration through firedispathrequest method. The server listens to the default port 10100 for incoming agents. The following is the list of some predefined methods of Boga Agents

- firedestroyrequest() - Destroys agent through process kill
- onCreate() - Called when an agent is created
- onArrival() - Called when an agent arrives on the base
- onDispatch() - Called when an agent moves from the base

The main CLASS : roamAgent consists of the following method definitions to perform the intended operations. They are defined as follows

- **onCreate () :** gets the initial IP of the machine to check for exe file CRCs under specified directory IFF the machine is reachable from the server.
- **onArrival () :** checks its own CRC before scanning files for suspicion using RMIJDBC connectivity. IFF it is valid, then collects exe files under the given directory using FILE NAME FILTER Interface. Check their CRC, reports to the server if any invalid entries detected.
- **run () :** jumps and executes scanning of files
- **doCheckSum () :** return the CRC of the windows executable file
- The roamAgent calls the mScan agent which scans the exe files (first 2 ^ 16 bits) against virus signatures (18 bit) stored in the database and returns the scan results to the user using **onArrival () and doCheckSum ()** methods.

5.3 Acquired Immuno Deficiency Syndrome

The Acquired Immuno Deficiency Syndrome (AIDS) of a Computer network is thought of as a collection of symptoms that do not have an easily identifiable cause since it is a condition that has to be contracted. The viruses and worms enter into the nodes and links and affect the entire network immunity at that part of the network that fights off viruses. The deficiency refers to resulting consequences that makes the immune system stop working properly. It is a syndrome because network with such an experience, any number of different types of symptoms and opportunistic malfunctioning or stoppage of the entire system will occur.

A file can be infected with worm or virus without developing AIDS. The virus can remain in computing node for many years without causing serious software or hardware problems. During this period, the virus is said to be latent, or inactive. Eventually, most computing systems which are infected do develop AIDS in distributed computing environment. Treatment of infected files or systems involves either trying to slow down or stop the virus from spreading into other nodes or files. During such a treatment, the prevention of other viruses that may enter into the immune network system can be reduced. The risk of contracting the virus increases if an individual node in the network has different illegal connectivity or practices and a number of unsafe browsing. Unsafe browsing refers to internet connectivity without using any method to prevent the exchange of spam and viruses without adhering to the stipulated security advises through firewalls and anti virus packages. An important element of the artificial immune system is like a group of white blood cells that include helper T cells, macrophages and monocytes which are the basic security and system management modules in the biological system. The immune system consists of all those spam preventions, software agents, and firewall mechanism that protect the node and/or network from infection by foreign machine code packets, such as worms.

6. PERFORMANCE AND RESULTS

The artificial immune network] is realized with five systems connected to a single server as a distributed system. The worm or virus attack is realized through the recent and wild virus signatures [11] of various sizes ranging from 32 to 512 bits. The files of different types like .exe, .zip and.com of different sizes are tested for the one or multiple virus infections in all the nodes concurrently. The performance of the proposed artificial immune network is evaluated on the basis of the total time taken including the time for creation, dispatch, travel, scanning and reporting the status of the files specified in the user defined target path at the destinations. The cost benefit factor, usability and extensibility of this approach are remarkable since from a single installation, information security of multiple nodes are achieved which can also be extended for future unknown viruses if their signature is identifiable. The sensitivity, specificity and the overall detection capability are the three performance metrics through which the evaluation of the work is carried out. Firstly the sensitivity of the proposed artificial immune network against worm or virus attacks depend on the number of nodes in the network since the roaming agent has to verify the check sum of the individual files. The numbers of virus signatures, nodes, files tested are 80, 5 and 35 respectively. The time taken for arrival, scan and reporting of bugs to the server by the roam and scan mobile agents are shown in Table 2.

Table 2. Scan and Report time for individual agents

Time for Creation/ Dispatch t_{cd} (sec)	**Travel time t_t**	**Time for CRC check**	**Total length of file machine code (bits)**	**Time for Scanning Virus Sign (sec)**	**Total time for Check (sec)**
6	9	6	64	-----	-----
3	9	-----	64	393	426
6	9	6	32	-----	-----
3	9	-----	32	237	270
6	8	5	24	-----	-----
4	9	-----	44	202	230

The connectivity parameters like the data transfer rate, intermediate firewalls and switches along with the bandwidth are also becoming the deciding factors. Keeping the above parameters constant, the sensitivity is calculated as the ratio of the time taken by the roaming agent to give the FIR to that of the total time covering the entire scanning process for a particular file. The sensitivity, specificity and detectability of the agent system are calculated using the travel, scan, report time as shown below. The size of virus signature database and number of nodes decreases the specificity factor and detectability of the agent respectively.

Table 3. Tested Worms ,Viruses and Trojans

Virus Name	Date discovered	Type	Reported Infections	Damage Potential	Signature
Rbot.210944	18/01/2006	Worm	Low	Medium	30d1ceeb713701948f1746dd3912f7d3
Mytob.MR	09/06/2004	Worm	Low	Medium	1c6c7cbb3e47fae15dd54da069cd5165
Womble.D	12/09/2006	Worm	Low	Medium	a7eed18c21897e50bbe167b8f438b9af
Virtumonde.267	02/04/2007	Trojan	Low	Low	731396df61f1cedc2b70ab33ebb0c0b3
NetSky.C	25/02/2004	Worm	Medium	Low	0e17dbec1904b7c10614bfb29ef758fd
Dldr.Nurech.BG	25/12/2005	Trojan	High	Low	041c6826efdea3f72b167ea7a535978c
SdBot.akv	17/11/2005	Worm	Low	Medium	e4c3dcd460c2e4c898c65a59161c2d80
Scano.Z	17/05/2006	Worm	Low	Low	b2f196575a081b0B3d0E3f4234b747ed
Banwarum.E	28/05/2006	Worm	Low	Medium	de42073eff6b9b12313915ad12c13bdb
Renchneg.B.Dl1	25/05/2006	Worm	Low	Medium	e78755206af1d523a79a0510e3106708
Agobot.97918	29/08/2005	Worm	Medium	Medium	445882B3C915350B29735DF1C8169ECB
Codbot.AP	29/08/2005	Worm	Low	Medium	c34b5ec44017814cb4b9718855267984

The Sensitivity Factor $= \dfrac{\mathbf{6 + 9 + 6}}{\mathbf{426 / 7}} = \mathbf{0.3450}$

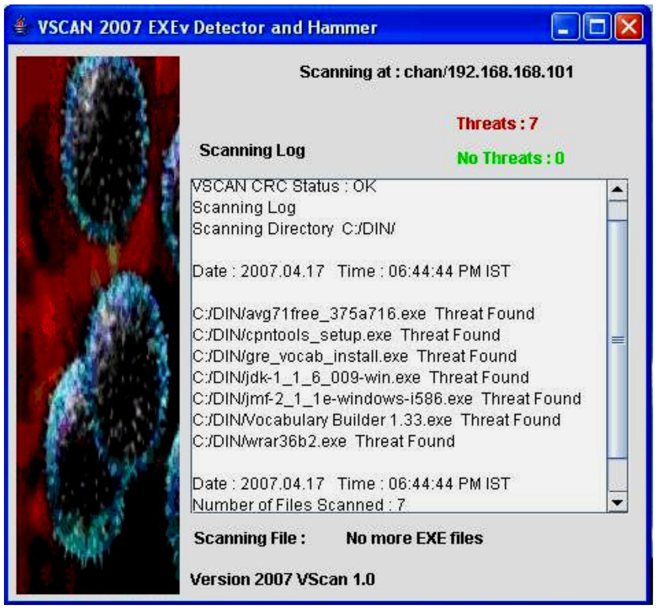
The Specificity Factor $=$ **64 / 80 * 64 = 0.0125**

The detectability of the AINS $= \dfrac{\text{Number of Signatures * size of signature}}{\text{Machine code size * No of nodes * No. of Files}}$

$=$ **80 * 64 / 65536 *5 * 7**

$=$ **0.002235**

With the proposed intelligent agent method it is possible to detect any attack with only a single point of control at the network server not at all nodes that will lead to a cost effective secured mechanism.



Scanning at : chan/192.168.168.101

Threats : 7

Scanning Log

No Threats : 0

```
VSCAN CRC Status : OK
Scanning Log
Scanning Directory  C:/DIN/

Date : 2007.04.17   Time : 06:44:44 PM IST

C:/DIN/avg71free_375a716.exe  Threat Found
C:/DIN/cpntools_setup.exe  Threat Found
C:/DIN/gre_vocab_install.exe  Threat Found
C:/DIN/jdk-1_1_6_009-win.exe  Threat Found
C:/DIN/jmf-2_1_1e-windows-i586.exe  Threat Found
C:/DIN/Vocabulary Builder 1.33.exe  Threat Found
C:/DIN/wrar36b2.exe  Threat Found

Date : 2007.04.17   Time : 06:44:44 PM IST
Number of Files Scanned : 7
```

Scanning File : No more EXE files

Version 2007 VScan 1.0

Figure8 Snapshot of the Implemented AINS using Agents.

8. CONCLUSION

The security of a computer or information network is enhanced by adopting the acquired immune behavior artificially in the distributed network. The immune network is modeled through a timed and event oriented Petri Net to decide the reachability and space complexity of such a system considering the similarities and the dissimilarities between the biological and artificial immune system. The network components including inherent system safety modules and the software agents for the quicker activation of the antibodies in the suspected location are achieved by intelligent agents. The remote virus database is searched for the suspected virus in terms of a malicious machine code and the correct java class file could be selected through a learning process. As an outcome of the decision made by the intelligent roaming agent, the scanning agent is in turn transferred to the suspected location as defined in the IP-address table. In the distributed environment, the mobile java class file is compiled and executed at the destination address and scans the relevant or specified files in the target path. Multiple copies of the scanning agents is created and dispatched to the various suspected nodes in the network depending upon the severity and multitudeness of the attack. The intelligent agent method is found to be working good for all types of viruses, worms and Trojans as tabulated in Table 3. The output GUI snapshot for the implemented technique is also shown in the Figure 8.
The intelligent agent model thus provides security in a distributed environment with minimum cost if all the nodes are in the same operating system. Again if multiple attacks are detected then suitable instances of agents from a single agent class are instantiated and packed up to the respective nodes concurrently. The self replicative model is implemented through java class files which are executed in the run time at the affected nodes.

8.1 Discussion

The proposed network model faces severe drawback in detecting the suspected worm due to the different operating system environment in a networking scenario. The speed of virus / worm detection is heavily depending upon the individual node's authentication and authorization settings. The ever increasing and alarming rate of unidentified intruders into a network makes the learning process more complex with multiple attributes of the antigen. The nature of attack by the viruses and their replication in very large number in a non deterministic manner makes the network susceptible to infection in a more complex manner. The correct and quicker selection of the mobile java class file tries to enhance the performance of the network but the sensitivity and specificity diminish due to the non availability of the exact virus signatures even though they are large in number but unknown with their abrupt time of attack. The over all detectability of AINS can be enhanced if some pre evidences of the style and origin of the attack were available.

9. REFERENCES

[1] Beer, R.D., Chiel, H.J. and Sterling, S., A Biological Perspective on Autonomous Agent Design, In Robotics and Autonomous systems, Vol. 6, (1990), 169 – 186.

[2] Dasgupta, D, Artificial Immune Systems and Their Applications, Heidelberg, Germany: Springer-Verlag, 1999.

[3] Dasgupta, D., An artificial immune system as a multi-agent decision support system, *Proc. IEEE Int. Conf. Systems, Man and Cybernetics* ,(Oct. 1998), pp. 3816–3820.

[4] David Kotz and Robert S. Gray, Mobile Agents and the Future of the Internet, *ACM Operating Systems Review,* (Aug. 1999), 7-13.

[5] Desel, J., and Reisig, W., Place/Transition Petri Nets. In Lecture on Petri nets I: Basic Models, vol 1491 of *Lecture Notes in Computer Science*, Springer - Verlag, 1998.

[6] Forrest S., Perelson A.S., Allen L., and Cherukuri, R., Self–Nonself Discrimination in a Computer, *Proceedings of the IEEE Symposium on Research in Security and Privacy*(Los Alamos, CA: IEEE Computer Society Press), 1994.

[7] Goel, S and Bush S.F., Biological Models of Security for Virus Propagation in Computer Networks login:, vol. 29, no. 6, (Dec. 2004), 49-56.

[8] Kaariboga Mobile Agents (Sep. 2003). [Online]. Available: http:// http://www.projectory.de/kaariboga/index

[9] Kephart, J.O., Biologically Inspired Defenses against Computer Viruses, *Proceedings of IJCA '95*, (1995) 985–996.

[10] Paul K. Harmer *et al*, An Artificial Immune System Architecture for Computer Security Applications, *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, (Jun. 2002), 252 – 280.

[11] Virus Information and Statistics, [Online]. Available: http:// http://www.avira.com/en/threats/