International Journal of Computer Applications (0975 - 8887)
Volume 179 - No.16, January 2018

Intelligent Intrusion Detection in Computer Networks using Swarm Intelligence

Apoorv Saxena
Department of Computer Science and Engineering
Jaypee University of Engineering and Technology, Guna
India

Carsten Mueller, PhD
Department of Computer Science and Engineering
Jaypee University of Engineering and Technology, Guna
India

ABSTRACT

Swarm Intelligence is inspired by the collective behaviour of many individuals. It is coordinated using decentralized control and self-organization. The individual simplicity and their complex group behaviours can outperform the vast majority of individual members when solving problems and making decisions. During recent years, the number of attacks on networks has dramatically increased and consequently, interest in network intrusion detection has increased among the researchers. In this research paper, a software architecture is modelled and implemented which uses Ant Colony Optimization (ACO), ACO is combined with Non-Negative Matrix Factorization method for classifying a computer network behaviour as a sequence of system calls.

General Terms

Swarm Intelligence, Intrusion Detection System

Keywords

Ant Colony Optimization, Meta-heuristics, Anomaly-based Intrusion Detection System, Non-Negative Matrix Factorization

1. INTRODUCTION

Unauthorized access to networks is currently one of the most serious threats to the hosting business. Intruders and viruses present the two biggest security threats to the industry. Firewalls and other simple boundary devices lack a degree of intelligence when it comes to observing, recognizing, and identifying attack signatures that may be present in the traffic they monitor and the log files they collect. Without sounding critical of such other systems capabilities, this deficiency explains why Intrusion Detection Systems (IDS) are becoming increasingly important in supporting to maintain proper network security. Whereas other boundary devices may collect all the information necessary to detect (and often, to foil) attacks that may be getting started or already underway, they havent been programmed to inspect for and detect the kinds of traffic or network behaviour patterns that match known attack signatures or that suggest potential unrecognized attacks may be incipient or in progress [17].

1.1 Intrusion Detection System

An IDS monitors network traffic and monitors for suspicious activity and alerts the system or network administrator. The IDS also respond to anomalous or malicious traffic by taking action such as blocking the user or source IP address or even port from accessing the network. Applications and operating systems around the world are full of security flaws on many levels. From the viewpoint of the traditional security paradigm, it should be possible to eliminate such problems through more extensive use of formal methods and better Software Engineering [18]. An IDS is a software architecture that implements the intrusion detection on a technical and procedural level. An Intrusion Prevention System (IPS) prevents unauthorized access incidents from being successful. To better protect the system from any attacks, Intrusion Detection and Prevention System (IDPS) [19] which provides a completely automated monitoring services, is deployed on the systems.

IDS comes in a variety of flavors and approaches the goal of detecting suspicious traffic in different ways. They are divided into Network-based (NIDS), Host-based (HIDS), Wireless-based (WIDS), Network Behaviour Analysis (NBA) and Mixed IDS (MIDS) intrusion detection system. Classification of intrusion detection systems based upon different types of approaches is visualized in Fig. 1. NIDS is placed at strategic points within the network to monitor traffic. Ideally you would scan all inbound and outbound traffic, however doing so might create a bottleneck that would impair the overall speed of the network, whereas HIDS are run on individual hosts or devices on the network and monitors the inbound and outbound packets from the device only and will alert the user or administrator of suspicious activity is detected.

WIDS is similar to NIDS, but it captures wireless network traffic, such as ad hoc networks, wireless sensor networks, and wireless mesh networks. Besides, an NBA system inspects network traffic to recognize attacks with unexpected traffic flows. Adopting multiple technologies as MIDS fulfills the goal for a more complete and accurate detection. There are IDS that monitor and alert and there are IDS that perform an action or actions in response to a detected threat. There are multiple IDS that detect based on looking for specific signatures of known threats-similar to the way antivirus software typically detects and protects against malware which is known as signature-based detection (SD). A signature is a pattern or string that corresponds to a known attack or threat. Because of using the knowledge accumulated by specific attacks and system

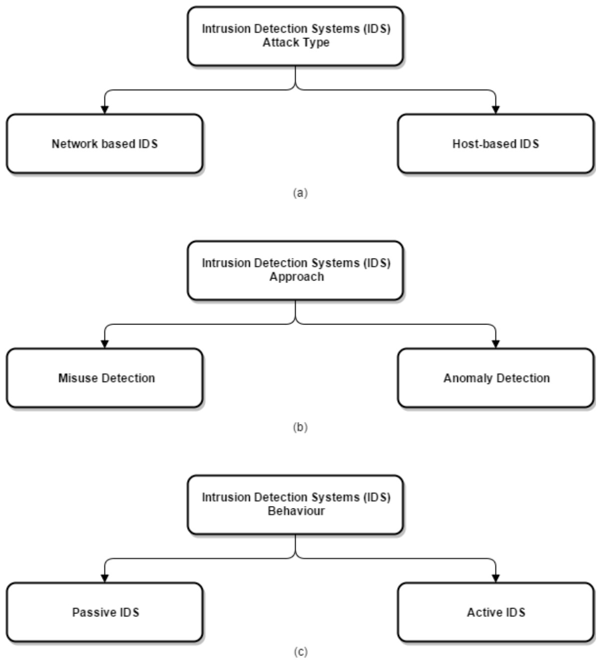
Intrusion Detection Systems (IDS) Attack Type
Network based IDS
Host-based IDS
(a)
Intrusion Detection Systems (IDS) Approach
Misuse Detection
Anomaly Detection
(b)
Intrusion Detection Systems (IDS) Behaviour
Passive IDS
Active IDS
(c)

Fig. 1. Classification of Intrusion Detection Systems [9] (a) Approaches based upon the type of attacks. (b) Approaches based upon the detection techniques. (c) Approaches based on the behaviour of IDS.

vulnerabilities, SD is also known as knowledge-based detection or misuse detection.

There are IDS that detect based on comparing traffic patterns against a baseline and looking for anomalies. An example of this type of IDS is anomaly-based IDS which will monitor network traffic and compare it against an established baseline. The baseline will identify an intrusion when the observed activities in computer systems demonstrate a large deviation from the norm profile built on long-term normal activities and it is able to detect even unknown attacks by comparing the current abnormal events with something that is considered normal. and alert the administrator or user when traffic is detected which is anomalous, or significantly different, then the baseline [2].

Furthermore, Stateful Protocol Analysis (SPA), also known as deep packet inspection when applied to networks, is a resource intensive approach to intrusion detection that, 'relies on vendor-developed universal profiles that specify how particular protocols should and should not be used' [10]. SPA provides important capabilities for understanding and responding to attacks. When the IDS are classified based upon the behaviour after the attack they come under active and passive IDS. An active IDS is also known as Intrusion Detection and Prevention System (IDPS). Intrusion Detection and Prevention System (IDPS) is configured to automatically block suspected attacks without any intervention required by an operator. IDPS has the advantage of providing real-time corrective action in response to an attack.

A passive IDS is a system thats configured to only monitor and analyze network traffic activity and alert an operator to potential vulnerabilities and attacks. A passive IDS is not capable of performing any protective or corrective functions on its own [20].

1.2 Ant Colony Optimization

Ant Colony Optimization (ACO) is a paradigm for designing Metaheuristic Algorithms for combinatorial optimization problems. Initially proposed by Marco Dorigo in 1992 in his Ph.D. thesis [1], a Metaheuristic is a set of algorithmic concepts that are used to define heuristic methods applicable to a wide set of different problems. Metaheuristic is a general-purpose algorithmic framework that is adapted to different optimization problems with minor modifications [15].

The inspiring source of ACO is the pheromone trail laying and following the behaviour of real ants which use pheromones as a communication medium. In analogy to the biological example, ACO is based on a colony of simple agents, called (artificial) ants, mediated by pheromone trails. The pheromone trails in ACO serve as a distributed, numerical information which the ants use to probabilistically construct solutions to the problem being solved and which the ants adapt during the algorithms execution to reflect their experience [16]. Pheromone encourages the following ants to stay close to previous moves. The pheromone evaporates over time to allow search exploration.

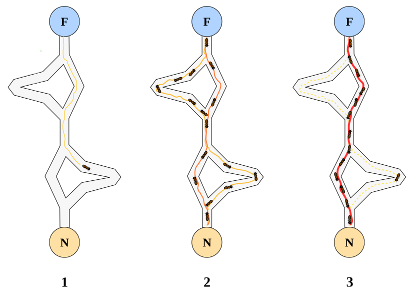

Fig. 2. Double Bridge Experiment [21] (1) Ants move between points F and N. (2) Find two path; Ants choose one of the two different paths with equal probability. (3) On the shorter path, more pheromone is laid down and is chosen for traversing.

In a number of experiments presented in [14], Dorigo and Stützle illustrate the complex behaviour of ant colonies. For example, In the first experiment, the bridge had two branches of different length one greater than another, the long branch was twice as long as the short one (see Fig. 2). At the start, ants were left free to move between the nest and the food source and the percentage of ants that chose one or the other of the two branches were observed over time. In this case, in most of the trials, after some time all the ants chose to use only the short branch, ants leave the nest to explore the environment and arrive at a decision point where they have to choose one of the two branches. Because the two branches initially appear identical to the ants, they choose randomly.

It is expected that, on average, half of the ants choose the short branch and the other half the long branch, although stochastic oscillations occasionally favor one branch over the other. The ants choosing the short branch are the first to reach the food and to start their return to the nest. When they must make a decision between the short and the long branch, the higher level of pheromone on the short branch will bias their decision in its favor. Therefore, pheromone starts to accumulate faster on the short branch, which will eventually be used by all the ants because of the autocatalytic process described previously. The difference between the two paths is called the preferential path effect; it is the result of the differential deposition of pheromone between the two sides of the obstacle, since the ants following the shorter path will make more visits to the source than those following the longer path. Because of pheromone evaporation, pheromone on the longer path vanishes with time [5].

ACO algorithm is the interplay of three central procedures (1) *ConstructAntsSolutions*, (2) *UpdatePheromones*, and (3) *DaemonActions*:

(1) *ConstructAntsSolutions* manages a colony of ants that concurrently and asynchronously visit adjacent states of the considered problem by moving through neighbor nodes of the problem's construction graph. They move by applying a stochastic local decision policy that makes use of pheromone trails and heuristic information.

(2) *UpdatePheromones* is the process by which the pheromone trails are modified. From a practical point of view, the deposit of new pheromone increases the probability that those components/connections that were either used by many ants or that were used by at least one ant and which produced a very good solution will be used again by future ants.

(3) *DaemonActions* is used to implement centralized actions which cannot be performed by single ants. Examples of daemon actions are the activation of a local optimization procedure, or the collection of global information that can be used to decide whether it is useful or not to deposit additional pheromone to bias the search process from a nonlocal perspective [5].

Algorithm 1 ACO Metaheuristic Algorithm

```
procedure ACO_METAHEURISTIC
    procedure SCHEDULEACTIVITIES
        ConstructAntsSolutions()
        UpdatePheromones()
        DaemonActions()
    end procedure
end procedure
```

In Algorithm 1, the ACO metaheuristic is described. The main procedure of the ACO metaheuristic manages the scheduling of the three procedures mentioned before via the *ScheduleActivities* construct: (i) management of the ants' activity, (ii) pheromone updating, and (iii) daemon actions. The ScheduleActivities construct does not specify how these three activities are scheduled and synchronized. It does not define whether they should be executed in a completely parallel and independent way, or if a kind of synchronization among them is necessary [5].

The paper propose an Anomaly Intrusion Detection System which uses Swarm Intelligence meta-heuristic Ant Colony Optimization (ACO). In this task, the goal is to use the artificial agents (ants) and find an optimal path between various process ID on the basis of heuristics and pheromone values so as to identify and detect intrusion inside a computer network.

The remainder of this paper is organized as follows. Section II introduces The Proposed Approach. Section III gives the Software Architecture of the proposed solution. Section IV follows the Statistical Analysis and Proof of Concept by using practical data and discusses the results and Finally, the Summary is discussed in the last Section V.

2. APPROACH

The paper proposes an intelligent model for intrusion detection system, the model follows an anomaly detection system that uses a novel method based on non-negative matrix factorization (NMF) to determine the anomaly index per process followed by a metaheuristic method of Swarm Intelligence known as ant colony optimization which is an optimization method used here to find and choose an optimized path between various processes and their anomaly indexes.

Non-negative matrix factorization (NMF) is a powerful data analytic technique which finds its applications in such fields as computer vision, document clustering, chemometrics, audio signal processing and recommender systems[11]. The salient feature of this method is its use of non-negativity constraints, leading to the parts-based representation and allowing only additive combination. This is specifically suitable for analyzing the frequency characteristics of individual system calls or commands since all the frequencies are positive [8].

Given an initial data set expressed by an $n \times m$ matrix X, where each column is an n-dimensional non-negative vector of the original data (m vectors). Based on the theory of NMF, it is possible to find two new matrices, W and H, to approximate the original matrix $X \approx WH$ [11](Shown in Fig. 3).

$$X_{i\mu} \approx (WH)_{i\mu} = \sum_{a=1}^{r} W_{ia} H_{a\mu} \tag{1}$$

where $X_{i\mu}$, W_{ia} and $H_{a\mu}$ are the elements of matrices X, W, and H, respectively.

The dimensionalities of the factorized matrices W and H are $n \times r$ and $r \times m$, respectively. Usually, r is chosen as a number smaller than n and m so that W and H are smaller than the original matrix X. Eq. (1) can be rewritten by column vectors as $x \approx Wh$, where x and h are the corresponding column vectors of W and H. In other words, each data vector x is approximated by a linear combination of the columns of W, weighed by the components of h. Therefore, W is regarded as containing a basis optimized for the linear approximation of X[11]. Each column of the matrix W contains basis vectors and each column of H contains encoding coefficients. The data matrix X is considered as original observation vectors, and the encoding matrix H as the feature vectors learned based on the basis W[8]. In order to find a factorization $X \approx WH$, one of the iterative approaches is given by the following rules[12]:

$$H_{a\mu} \leftarrow H_{a\mu} \frac{(W^T X)_{a\mu}}{(W^T W H)_{a\mu}} \tag{2}$$

$$W_{ia} \leftarrow W_{ia} \frac{(X H^T)_{ia}}{(W H H^T)_{ia}} \tag{3}$$

The anomaly intrusion detection model implemented in this paper was originally proposed by Xiaohong Guan, Wei Wang and Xiangliang Zhang (2009)[8] which is based on NMF method. But instead of using the transition information of the data in hidden Markov models based on short sequences of continuous system calls by [6, 13, 22], etc or Markov chain models based on command sequences by [23, 3] etc., they have used the frequency properties of system calls and commands to analyzed the characterization of program and user behaviours, respectively. The steps to building this intrusion detection model include data preparation, reduction and feature extraction by NMF and intrusion detection[8].

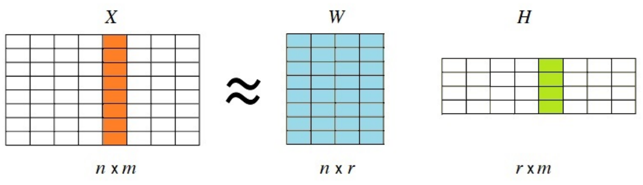

Fig. 3. Non-Negative Matrix Factorization

In the data preparation step, the original data set or input data which consists of collections of processes and their associated system calls is being prepared and transformed into smaller data blocks. Grouping of system calls and processes occurs among system calls which are associated with the same process and thus transformed into one. Along with the frequency of each individual system call in each trace is calculated at the same step. The reduction and feature extraction takes place on this reduced data block of original data and is computed against the trained data with iterative updating algorithm in Eq. (2) and Eq. (3) which is proposed by the NMF to generate a new form of data which represents the features. At the last step, these features are used to tell whether the intrusion is detected a process or not. If the data vector corresponds to normal behaviours, the sum of all the elements in the data vector should be approximately equal to number 1, if not then it is treated as anomalous and it can be said that intrusion has occurred and an anomaly index is generated against the process denoting the deviation it shows from the normal behaviour and to show that intrusion has occurred in it.

To make this system more intelligent, efficient and effective the paper proposes an ant colony optimization module on top of this anomaly intrusion detection system by Xiaohong Guan, Wei Wang and Xiangliang Zhang (2009)[8]. It uses the result information generated by the NMF at the intermediate stages inside ACO metaheuristics. The ant colony optimization follows the combination of two-pronged approaches towards the solution, one is heuristics based and another one is pheromone based. The heuristics approach are problem specific, each problem has its own heuristic approach toward solving the problem and hence, they cannot be generalized. For example, the heuristic approach described by Marco Dorigo [4] for the travelling salesman problem (TSP) uses heuristic information η_{ij}, which is typically inversely proportional to the distance between cities i and j, a straightforward choice being $\eta_{ij} = 1/_{d_{ij}}$. Whereas the pheromone approach is more general purpose and can be applied to series of the problem it generally follows pheromone updating step, Once every ant has constructed an assignment, each pheromone trail of the pheromone structure is decreased and then the best ants of the cycle deposit pheromone. Both the parameters play very important role in an ACO algorithm, the heuristic information is very important in generating high-quality tours in the initial search stages. Because the value of the pheromone trails do not have much information in the early stage of learning and cannot guide the artificial ants in constructing good tours. On the other hand, in the later stage, the pheromone trails may collect enough information to behaviour as required and the heuristic information now won't play very much of a part in tour generation due to its locality.

The Ant System which was introduced by Marco Dorigo [1] provides a system with multiple artificial ants moving around in a graph-like data structure. In this system, the ants play the role of an agent which initially starts at a random position and at each step iteratively adds one still unvisited process to its partial tour. In order to construct a feasible solution, the ants select the process to be visited next on the basis of a probabilistic decision rule, this probabilistic decision takes into consideration both the intensity of trails between edges with a pheromone value represented by τ and the heuristic visibility between edges represented by η which is generated through a heuristic function and these two parameters are being controlled by two adjustable positives α and β. As each of the ants completes its tour by visiting all the processes, the pheromone amount on each path generated by these ants is adjusted with the process of pheromone update through Eq. (4).

$$\tau_{xy} \leftarrow (1-\rho)\tau_{xy} + \sum_{k} \Delta\tau_{xy}^{k} \tag{4}$$

where τ_{xy} is the amount of pheromone deposited on the Matrix X generated from the original data here x and y denotes PID and System Call ID, ρ is the pheromone evaporation coefficient and Δ τ_{xy}^{k} is the amount of pheromone deposited by kth ant.

$$\Delta\tau_{xy}^{k} = \begin{cases} Q/L_k & \text{if ant } k \text{ uses curve } xy \text{ in its tour} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where L_k is the cost of the kth ant's tour (total anomaly index in our cases) and Q is a constant (typically Max Deviation that can occur).

In general, the kth ant moves from process x to process y with probability

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z\in \mathrm{allowed}_x}(\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})} \tag{6}$$

where τ_{xy} is the amount of pheromone trail on edge x, y, $0 \leq \alpha$ is a parameter to control the influence of τ_{xy}, $\beta \leq 1$ is a parameter to control the influence of η_{xy}. τ_{xz} and η_{xz} represent the attractiveness and trail level for the other possible state transitions. Here η_{xy} is the desirability, a prior knowledge, a heuristics function which is given by Eq. (7).

$$\eta_{xz} = Q/H \tag{7}$$

where H is the anomaly index of the process. Each ant constructs a complete route and at each construction step, the ant applies the probabilistic rule using Eq. (6). The implementation of the probabilistic rule requires some care to avoid large computation times. The procedure works as follows: first, the current process of ant is determined after that the probabilistic choice of the next process then works analogously to the roulette wheel selection procedure of evolutionary computation [7]: each process that ant has not visited yet determines a slice on a circular roulette wheel, the size of the slice being proportional to the weight of the associated choice. Next, the wheel is spun and the process to which the marker points is chosen as the next process for ant [5].

3. SOFTWARE ARCHITECTURE

In order to dive deeper, a class diagram is prepared so as to get a better view towards the software architecture of the proposed model for intrusion detection in computer networks (Shown in Fig. 4). The class diagram provides a static view of the system which acts as a base component in designing the model, the class diagram of the model proposed by the paper consists of multiple classes working together to achieve the same target of detecting intrusion. The different classes in the class diagram of the proposed model are Application, IDSLoader, IDS, FrequencyMapper, Ant Colony,

Table 1. Example of sequence of system calls

Process ID	System Call ID
2867	246
2867	230
⋮	⋮
3036	246
3036	117
⋮	⋮
3106	267
3106	117
⋮	⋮

Ant, MersenneTwister, and Configuration.

The Application class is the main class of the system. IDSLoader class role in the implementation of the model is to handle the task of data preparation. It loads the observation data into the system from the raw data file containing a long queue of the sequence of system calls associated with their process. An example of the sequence of system calls generated by various processes are shown in Table 1.

In this sequence, each system call is represented by a number. The mapping between a system call number and the actual system call name is given by a separate table. For example, the number "246" represents system call "sys_io_destroy", the number "6" represents system call "sys_close". The above example is a very short sequence of system call the actual data contains thousands of system calls per process, So in order to compute faster, these data are then further divided into smaller blocks of data. Now here comes FrequencyMapper class in action, it moulds the data into a data vector of a matrix form by computing the frequencies of individual system call associated with a process using Eq. (8).

$$f(n) = \frac{RespectiveSysCall(n)Frequency}{TotalNumberOfFrequency} \tag{8}$$

Each trace of system call data, invoked by each process, is thus transformed into a data vector and the matrix representing a system call data set is generated as shown in Table 2.

Table 2. Trace of System Call

Distinct System Call	Process ID				
	5553	3106	…	5421	3918
1	0.0016	0.0000	…	0.0316	0.0000
2	0.0000	0.0000	…	0.0000	0.0000
⋮	⋮	⋮	⋮	⋮	⋮
336	0.0000	0.0000	…	0.0000	0.0000
337	0.0035	0.0000	…	0.0023	0.0000
⋮	⋮	⋮	⋮	⋮	⋮
400	0.0000	0.0000	…	0.0000	0.0000

The above matrix represents a smaller data block which consists of many other smaller block of data, transformed from a bigger observational data set. In this way, the observational data set is divided into m blocks of data each containing a total of n distinct observations which are actually system call frequencies. Now each block of data is represented by a vector of length n denoted as

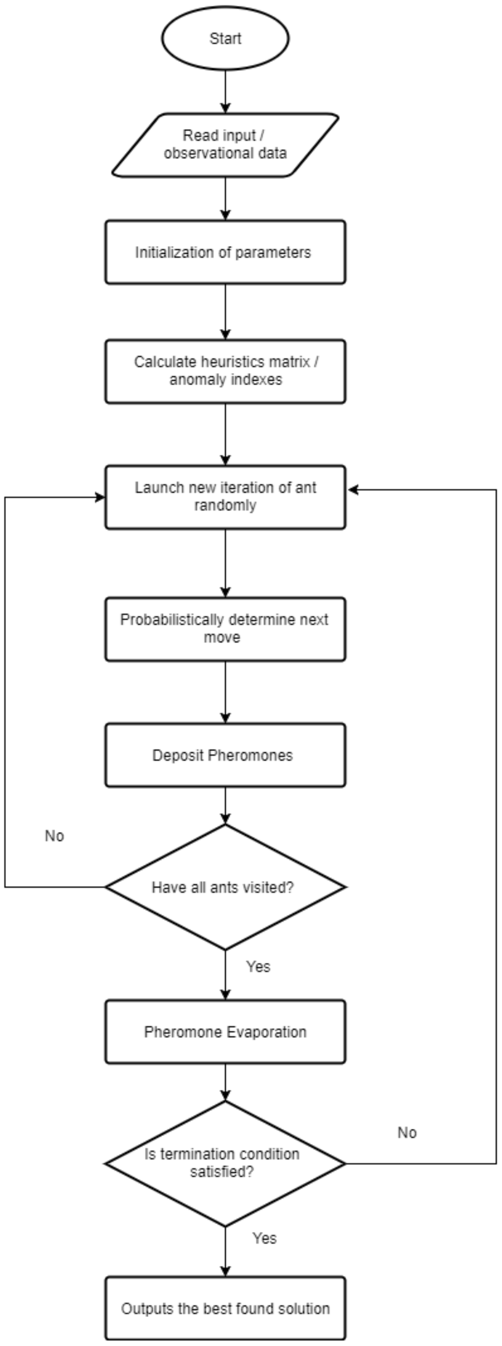
Start
Read input / observational data
Initialization of parameters
Calculate heuristics matrix / anomaly indexes
Launch new iteration of ant randomly
Probabilistically determine next move
Deposit Pheromones
No
Have all ants visited?
Yes
Pheromone Evaporation
No
Is termination condition satisfied?
Yes
Outputs the best found solution

Fig. 5. Sequence of flow control in the proposed approach

$\mathbf{f_i}$. A $n \times m$ matrix X, where each element X_{ij} stands for the frequency of i^{th} distinct observation occurs in the j^{th} block, is then constructed. The observed data set that is represented by a matrix $X_{n\times m}$ can be written as:

$$M_{n\times m} = \begin{pmatrix} f_{11} & f_{12} & \cdots & f_{1m} \\ f_{21} & f_{22} & \cdots & f_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n1} & f_{n,2} & \cdots & f_{nm} \end{pmatrix} = \begin{bmatrix} \mathbf{f_1} & \mathbf{f_2} & \cdots & \mathbf{f_m} \end{bmatrix} \tag{9}$$

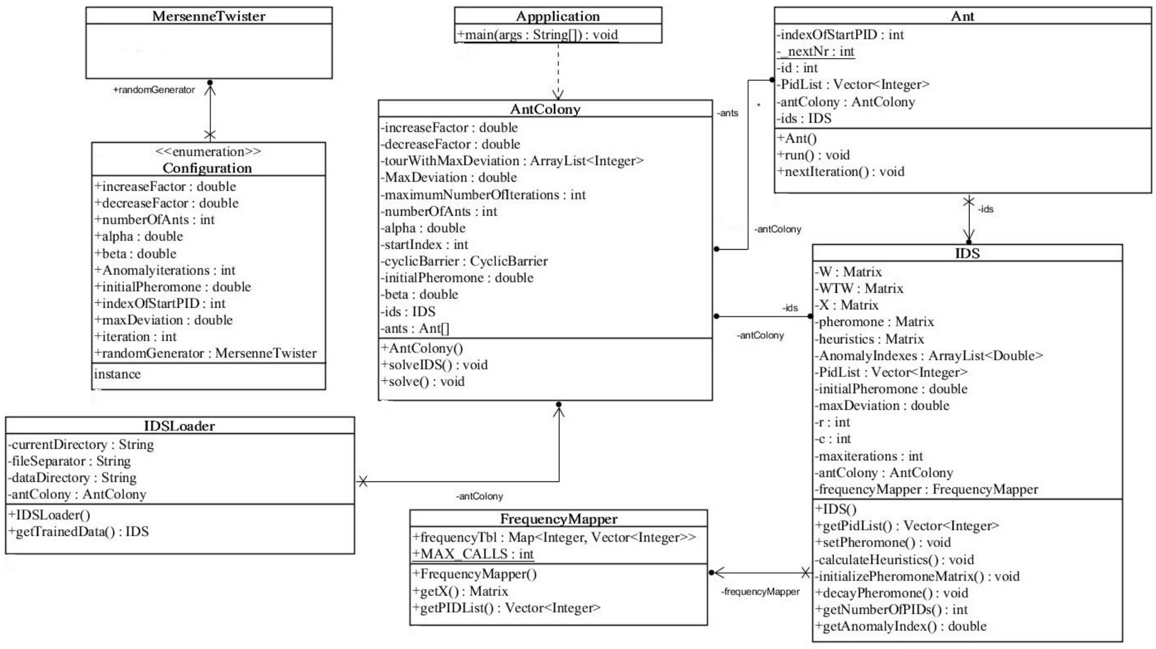

Fig. 4. Broad overview of class diagram of the proposed approach towards Computer Intrusion System

where vectors $\mathbf{f_1}, \mathbf{f_2}, \cdots, \mathbf{f_m}$ represent the corresponding blocks of the original data. Clearly based on the frequency property, the following normalization condition is satisfied[8]:

$$\sum_{i=1}^{n} X_{ij} = 1, \qquad j \in \{1, 2, 3, ..., m\} \tag{10}$$

Now when the data is successfully been prepared for the system, then it is passed on further to other different classes of the system to be computed upon. Initially, it passed through different classes via IDS class, the IDS class plays a very essential and important role in the Ant Colony Optimization algorithm as it is responsible for commencing and managing different important initialization tasks of an ACO algorithm such as setting heuristics matrix, setting pheromone matrix, calculating heuristics and finally monitoring the decaying pheromone process which executes after an ant successfully traverse a tour. Initialization of heuristics and pheromone matrix takes place at this class. The pheromone matrix and heuristics matrix are made with the same dimensions as that of the matrix X generated at the data preparation stage in IDSLoader class and are initialized in the same class. The initialization of pheromone matrix is very easy since it take's in only constant value for all the elements at the start then the values of the matrix becomes dynamic after the ant's started constructing solutions, but the same cannot be said about heuristics matrix it requires a prior knowledge towards the problem which it fetches through a heuristics function given as Eq. 7 and its value remains static throughout the execution of the system.

The heuristics function Eq. 7 uses the anomaly index of a process, which is generated through non-negative matrix factorization method proposed by [8]. Now the data block generated by IDSLoader is fetched and NMF method is applied to it to firstly reduce the data block and find out its features in order to generate an anomaly index. The block of data in data vector form consists of m blocks of training data, the data set can be represented by vectors $\mathbf{f_1}, \mathbf{f_2}, \cdots, \mathbf{f_m}$ or matrix $X_{n\times m}$. With the iterative updating algorithm given by Eq. 2 and Eq. 3, the training data set represented by $X_{n\times m}$ is factorized into the $n \times r$ bases W and $r \times m$ coefficients H. The matrix H represents the features learned from the original matrix X and the features associated with each block of data are represented by column vector h in H. The iterative updating algorithm in Eq. 2 and Eq. 3 has the following property:

$$\sum_{i=1}^{r} H_{ij} = \sum_{i=1}^{n} X_{ij}, \qquad j \in \{1, 2, 3, ..., m\} \tag{11}$$

Based on Eq. 10 and Eq. 11 can also be written as

$$\sum_{i=1}^{r} H_{ij} = 1, \qquad j \in \{1, 2, 3, ..., m\} \tag{12}$$

From Eq. 12 we can conclude that if the summation of all the elements in each column vector $\mathbf{h}$ of H equals to 1, then it can be classified as of normal behaviour or non-anomalous profile. So given a data vector t that represents a block of data obtained in the real-time, then a coefficient vector $\mathbf{h_t}$ can be obtained through iterative rule Eq. 2 based on the basis W learned from the training data set. If the data vector t corresponds to normal behaviours, its associated coefficient vector should have similar characteristics with those coefficient vectors learned from training data vectors. In other words, if the data vector t corresponds to normal behaviours, the sum of all the elements in $\mathbf{h_t}$ should be approximately equal to the number 1. Let $d = \sum_{i=1}^{r} h_{t_i}$ and a simple classifier is defined as the anomaly index for intrusion detection

$$|(d-1)| = \varepsilon \tag{13}$$

If ε is above a threshold, the block of data associated with the data vector t is then considered as normal. Otherwise, it is treated as anomalous with an anomaly index ε. After all the anomaly index of per process has been evaluated it is then distributed in the heuristics matrix in a column wise fashion using Eq. 7. Hence a prior knowledge is prepared and laid on the matrix along with initial pheromone value. Now, we can commence our ACO algorithm since we have all the prerequisite prepared and also initialized all the matrices and other algorithm parameters. The majority part of the algorithm focuses on the construction of the solution as shown in Algorithm 1 followed by updating of parameters and matrices based on the path created by ant. The AntColony class takes care of the monitoring and initialization of the algorithm it also helps creates artificial ants in the system using another class Ants which extends Threads thus providing it with an ability to make multiple ants and perform concurrently in the environment. The ants in the system are responsible for the construction of tour and they do so in phases one by one. Initially, ants memory should be null thus marking all the processes as unvisited. After that, the ants require an initial process to be assigned to it so that it can construct a tour starting from that process, this can be done manually or even through a random process assignment function. Next, comes the actual tour construction by each ant through a series of construction steps which requires care since ant located at one process choose to move to another probabilistically on the basis of pheromone count and heuristic value associated with it. The probabilistic choice of the next process then works analogously like roulette wheel selection procedure, each process that ant has not visited yet determines a slice on a circular roulette wheel. Next, the wheel is spun and the process to which the marker points is chosen as the next process to be visited for ant. The whole idea of roulette wheel selection is done in a sequenced manner. First, the weights of all the various choices that are selection probabilities of the processes are determined using following equation:

$$selection\_probability = [\tau_{xy}]^{\alpha}[\eta_{xy}]^{\beta} \tag{14}$$

where τ_{xy} is the amount of pheromone trail on edge x y and η_{xy} is the desirability, a prior knowledge. Here $0 \leq \alpha$ is a parameter to control the influence of τ_{xy}, $\beta \leq 1$ is a parameter to control the influence of η_{xy}.

After that these values are stored into a variable $sum\_probabilities$, now a random number r is picked up from the uniformly distributed interval $[0, sum\_probabilities]$ then we go through the feasible choices until the sum is greater or equal to r. Finally, the ant moves to that process and all these construction steps are repeated again and again until all the ants have completed a tour. Since each ant has to visit the same number of processes, all the ants complete the solution construction after the same number of construction steps.

Once the solutions are constructed, the last step in an iteration is to update the pheromone value associated with the tour as shown in Eq. 4. This function comprises two pheromone update procedure. One is handled by the Ant class which is pheromone deposit and another is pheromone decay which is handled at the IDS class. The deposit pheromone procedure deposits the pheromone thus adds pheromone to the arcs belonging to the tours constructed by the ants and the decay pheromone process does quite the opposite of this. it decreases the value of the pheromone trails on all the arcs by a constant factor ρ. But the programs stop if a termination condition is satisfied, there can exist several termination conditions some of the possible termination conditions are: (1) the algorithm has found a solution within a predefined cost from an upper bound on the optimal solution quality; (2) maximum number of iterations has been reached; (3) a maximum amount of CPU time has been spent; or (4) the algorithm shows stagnation behavior. To have a greater insight of the proposed system a flow chart is prepared and is presented in Fig. 5 which show the work flow of process and the sequence of steps and decisions needed to perform in order to achieve the proposed system.

4. STATISTICAL ANALYSIS AND PROOF OF CONCEPT

The solution quality of the IDS using Swarm Intelligence is determined by a high detection rate and minimum deviation related to the training data. This section examines with a statistical analysis the influence of the two central parameters α and β on the solution quality.

(1) α: The parameter α is used for controlling the importance of pheromones while calculating the transition probabilities. A value of $\alpha = 0$ indicates that ants ignore their global knowledge, which turns the algorithm into a greedy heuristic. Typical values are therefore $\alpha \geq 1$.

(2) β: The parameter β regulates the significance of the heuristic component instead of the pheromones. The setting $\beta = 0$ causes the ants only to use their accumulated knowledge. Appropriate values are $\beta \geq 1$.

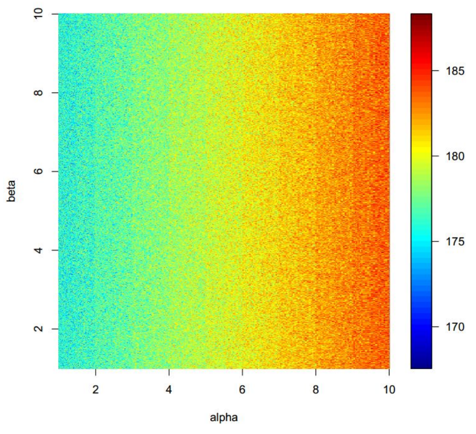

Fig. 6. Parameter test with 15 ants, increase factor equals to 0.05, decrease factor equals to 0.3, initial pheromone equals to 0.01 and in a range for alpha from 1 to 10 with a step size of 0.01, beta from 1 to 10 with a step size of 0.01.

The following statistical analysis (see Fig. 6) shows the successful implementation and proof of concept of this new approach which combines the Non-negative Matrix Factorization (NMF) algorithm for profiling a program user behaviours for anomaly intrusion detection and Ant Colony Optimization algorithm (ACO) for solving computational problems by a probabilistic technique for a productive intelligent intrusion detection system.

5. SUMMARY

The paper presents a modern intelligent model towards an ancestral problem of security threats in computer network system. There exists a number of intrusion detection techniques, but the paper uses an anomaly based detection technique which can detect an unknown threat towards the system while minimizing the time and cost of manual intervention and human error. In general, intrusion detection is a pattern recognition problem which includes two phases, feature selection and extraction and, then classification in different profiles on the basis of features extracted.

This paper proposes a new approach which combines the Non-negative Matrix Factorization (NMF) algorithm for profiling a program and user behaviours for anomaly intrusion detection and Ant Colony Optimization algorithm (ACO) for solving computational problems by a probabilistic technique to make one intelligent intrusion detection system. The Non-negative Matrix Factorization algorithm includes various steps in building intrusion detection model, this includes dividing the massive original data into small blocks of data and mapping the frequencies of occurrence of each individual elements in each block of data to construct a more simplistic matrix so that a large amount of data can be represented and reduced to a smaller processable data, then features from each block are extracted and factorization takes place to represent the blocks of data by a single number which makes it easier for them to classify normal program and user behaviours into anomalous and non-anomalous profiles without any additional classifier. These single number values of the data block are the anomaly indexes of the processes. Now that anomaly indexes have been generated for each process Ant Colony Optimization algorithm can be implemented, it runs on the top of data generated through NMF algorithm. The ACO deploys artificial agents/ants in the system which traverses through processes in order to generate an optimal feasible path. The ants use a probabilistic decision rule to select the next process to be iteratively added to the solution, this probabilistic decision takes into consideration both the intensity of trails (pheromone value) and the heuristic visibility a prior knowledge (anomaly indexes), both contribution in the environment is controlled by two adjustable parameters, this procedure works analogously to the roulette wheel selection procedure where the wheel is spun and the process to which the marker points is chosen as the next process for ant.

The model inherent parallelism allowing it have a really low over head, it also has a wide applicability since it can process of kinds of data even with the mixture, system call, UNIX commands, etc of large size. One of the significant benefits of anomaly-based detection methods is that they can be very effective at detecting previously unknown threats. The combination of NMF and ACO gives it the ability to rapidly discover good solutions making it easy to use as a dynamic application used in real-time intrusion detection system. There also exists some disadvantages to the model proposed since it uses NMF for profiling, then if frequencies of system calls or instructions generated by an unfriendly program or an unauthentic user are very alike to those produced by regular programs or authentic users though the sequences are quite different, NMF can hardly detect the anomaly. Also, there exists uncertainty in ACO as the time of convergence is unknown.

The method is implemented and tested and the numerical results show that the performances of the proposed model are promising and is a strong competitor to other well-developed heuristic algorithms. Authors further work will focus on a comprehensive study of the algorithm parameters and the simplification of the computational process. However, there is still much that remains to be explored as to the best of our knowledge, there are only a few applications of ACO algorithms including the proposed one in intrusion detection system. Therefore, the authors hope that this paper constitutes an important starting point for addressing Research and Development and will promote the study in the fields of ACOs and IDS among fellow researchers in the development of ACO based algorithms with excellent robustness and convergence.

6. REFERENCES

[1] Vittorio Maniezzo Alberto Colorni, Marco Dorigo. Distributed Optimization by Ant Colonies. In *European Conference on Artificial Life*, pages 134–142, 1991.

[2] Tony Bradley. Introduction to intrusion detection systems (ids).

[3] Sung-Bae Cho and Hyuk-Jang Park. Efficient anomaly detection by modeling privilege flows using hidden markov model. *Comput. Secur.*, 22(1):45–55, January 2003.

[4] M. Dorigo and L. Gambardella. Ant Colonies for the Traveling Salesman Problem. *BioSystems*, 43:73–81, 1997.

[5] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. Bradford Company, Scituate, MA, USA, 2004.

[6] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. A sense of self for unix processes. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, SP '96, pages 120–, Washington, DC, USA, 1996. IEEE Computer Society.

[7] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[8] Xiaohong Guan, Wei Wang, and Xiangliang Zhang. Fast intrusion detection based on a non-negative matrix factorization model. *J. Netw. Comput. Appl.*, 32(1):31–44, January 2009.

[9] Navneet Kaur Harshna. Survey paper of fuzzy data mining using genetic algorithm for intrusion detection. *International Journal of Scientific & Engineering Research*, 4(6), jun 2013.

[10] Peter M. Mell Karen A. Scarfone. Guide to intrusion detection and prevention systems (idps). Special Publication 800-94, NIST, Gaithersburg, MD, February 2007.

[11] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401:788–791, 1999.

[12] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 556–562. MIT Press, 2001.

[13] Wenke Lee and Salvatore J. Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th Conference on USENIX Security Symposium - Volume 7*, SSYM'98, pages 6–6, Berkeley, CA, USA, 1998. USENIX Association.

[14] A. Colorni M. Dorigo, V. Maniezzo. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26(1):29 – 41, February 1996.

[15] Luca M. G M. Dorigo. Ant Colony system: A Cooperative learning approach to the Travelling Salesman Problem. *IEEE*

transaction on evolutionary computation, 1(1):53 – 66, April 1997.

[16] Thomas Sttzle Marco Dorigo. The ant colony optimization metaheuristic: Algorithms, applications, and advances. In *Handbook of Metaheuristics*, pages 250–285. Springer US.

[17] Robert J. Shimonski. What you need to know about intrusion detection systems, Nov 2002.

[18] Anil Somayaji. Steven A. Hofmeyr, Stephanie Forrest. Intrusion Detection using Sequences of System Calls. *Journal of Computer Security*, 6(3):151 – 180, August 1998.

[19] Shamik Sural Suvasini Panigrahi. Detection of database intrusion using a two-stage fuzzy system. In *Information Security*, pages 107–120. Springer Berlin Heidelberg.

[20] Jajish Thomas. Types of intrusion detection systems (ids).

[21] M. Duran Toksari. A hybrid algorithm of ant colony optimization (aco) and iterated local search (ils) for estimating electricity domestic consumption: Case of turkey. *International Journal of Electrical Power and Energy Systems*, 78:776 – 782, 2016.

[22] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: alternative data models. *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, pages 133–145, 1999.

[23] Dit-Yan Yeung and Yuxin Ding. Host-based intrusion detection using dynamic and static behavioral models. *Pattern Recognition*, 36:229–243, 2003.