Available online at sjuoz.uoz.edu.krd

Science Journal of University of Zakho

Vol. 5, No. 4, pp. 313 –318, Dec.-2017

p-ISSN: 2410-7549
e-ISSN: 2414-6943

DIDS BASED ON THE COMBINATION OF CUTTLEFISH ALGORITHM AND DECISION TREE

Adel S. Eesa[a]*, Adnan M. A. Brifcani[b], Zeynep Orman[c]

[a] Dept. of Computer Science, Faculty of Science, University of Zakho, Kurdistan Region - Iraq (adel.eesa@uoz.edu.krd).
[b] Presidency of Duhok Polytechnic University, Kurdistan Region – Iraq.
[c] Dept. of Computer Engineering, Faculty of Engineering, Istanbul University, 34320, Avcilar, Istanbul, Turkey

Received: Jul. 2017 / Accepted: Dec., 2017 / Published: Dec., 2017 *https://doi.org/10.25271/2017.5.4.382*

ABSTRACT:
Different Distributed Intrusion Detection Systems (DIDS) based on mobile agents have been proposed in recent years to protect computer systems from intruders. Since intrusion detection systems deal with a large amount of data, keeping the best quality of features that represent the whole data and removing the redundant and irrelevant features are important tasks in these systems. In this paper, a novel DIDS based on the combination of Cuttlefish Optimization Algorithm (CFA) and Decision Tree (DT) is proposed. The proposed system uses an agent called Rule and Feature Generator Agent (RFGA) for reducing the dimensionality of the data by generating a subset of features with their corresponding rules. RFGA agent uses CFA to search for optimal subset of features, while DT is used as a measurement on the selected features. The proposed model is tested on the KDD Cup 99 dataset. The obtained results show that the proposed system gives a better performance even with a small subset of 5 features when compared with the using all 41 features.

KEYWORDS: Feature Selection Distributed Intrusion Detection System, Cuttlefish Optimization, Mobile agent.

1. INTRODUCTION

An Intrusion Detection System (IDS) is a security mechanism that gathers both user and system operations in computer and network systems and processes these information in order to determine the intrusive and attacker's events and can take some actions to abort these dangerous events (Center, 2009). A new approach (Steven R. Snapp, 1991) is the development of DIDS, where sensors (host and network based) gathers data, pre-process it and send it to a centralized station which is able to analyse and process this input.
Recently, a new paradigm on the development of DIDS which is based on Mobile Agents (MA) has attracted many researchers (Donald G. Marks, 2004; E., 2005; Imen Brahmi, 2010; Manmeet S, 2007; R. Sasikumar, 2012). MA is a composition of a software program and the data that can be defined as an autonomous program which is able to migrate and move from one node to another. It is commonly featured with autonomy, social ability and learning (Saidat Adebukola Onashoga, 2009). Keeping the best quality of features that represent the whole data and removing the noisy features, is an important function in IDS which can perform on the accuracy rate and computation time. In their previous study (Z. O. Adel Sabry Eesa, Adnan Mohsin Abdulazeez Brifcani, 2015), proposed a new feature selection model based on the combination of CFA and DT to reduce the dimensionality of the IDS dataset. This motivates us to reuse this model as an agent for designing a new DIDS in distributed environment. Some related previous studies in the literature can be found in (Chi-Ho Tsang, 2007; Hai Thanh Nguyen 2010; Jean-Louis Lassez 2008; Mohanabharathi R, 2012; N. Pratik Neelakantan 2011; Rupali Datti, 2012; Shih-Wei Lin, 2012; V. Bolón-Canedo, 2011).
This paper falls into 6 sections, as follows: Section 2 presents a brief introduction to MA, CFA and DT. The architecture, rule producer, and the mechanism of feature selection of the proposed system are discussed in detail in Section 3. Section 4 describes evaluation criteria for the proposed system. While section 5 highlights the experimental results and the discussion on the results reached. Lastly, a conclusion and some future works are listed in section 6.

2. BACKGROUND

2.1. Agents

Agent can be defined as an entity that can perform some tasks independently without any supervision. It can adapt itself, make decisions and collaborate with other agents. Besides working independently, the agent can perform some tasks with other agents and interact with the change of the environment ("The Agent-Oriented Software Engineering Handbook," 2004). There are two types of agents static and mobile agents (Wang J., 2006).
Static agent is firstly applied in the field of intrusion detection as an agent technique. It is the agent that remains in a fixed position or some fixed platforms, while mobile agent is an agent that can migrate from one node to another through the network. It could be dynamically distribute on the server interfaces which can be monitored on different sites. It also ensures a big level of resistance to network breakdowns and provides bandwidth saving since communication between mobile agent and the server, only takes part in locally exchanged messages which are not passed by the network (Dalila Boughaci, 2006).

2.2. Cuttlefish Optimization Algorithm CFA

CFA is a new optimization algorithm (A. M. A. B. Adel Sabry Eesa, Zeynep Orman, 2013). It is simulates the mechanisms used by cuttlefish to change its color. The shapes and colors seen in the skin of the cuttlefish are generated by the six cases of light reflection. The mechanism of light reflection is due to

* Corresponding author

the collaboration of different layers of cells including (chromatophores, leucophores and iridophores) as shown in Figure 1.

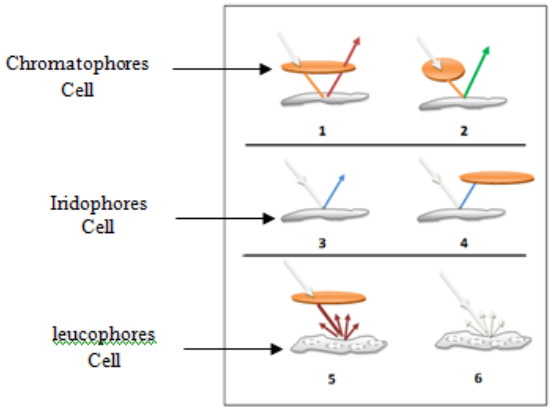

Figure 1. six cases of reflected light

Two main operations are designed for this algorithm the reflection operation and the visibility operation. Reflection is designed to mimic the mechanism of reflected light, while the visibility operation is designed to mimic the visibility of matched patterns. These two operations are designed to calculate the new solution or new point as in Equation 1.

$$newp = reflection + visibility \quad (1)$$

The main steps of CFA are described in Algorithm 1. The equations considered in Algorithm 1 are given as follows:

Case (1,2):

$$reflection_j = R * G_1[i].Points[j] \quad (2)$$

$$visibility_j = V * (Best.Points[j] - G_1[i].Points[j]) \quad (3)$$

Case (3, 4):

$$reflection_j = R * Best.Points[j] \quad (4)$$

Case (5):

$$reflection_j = R * Best.Points[j] \quad (5)$$

$$visibility_j = V * (Best.Points[j] - AV_{Best}) \quad (6)$$

Where G_1 is a group of cells, i, is the i^{th} cell in G_1,$Points[j]$ is the j^{th} point of i^{th} cell, *Best.Points* represents the best solution points, R represents the rate of reflection, V represents the visibility rate of the general appearance of the pattern, while the AV_{Best} is the mean value of the *Best* solution points. R and V are found as follows:

$$R = random() * (r_1 - r_2) + r_2$$
$$V = random() * (v_1 - v_2) + v_2$$

Where, *random* is a function which is used to produce some small random numbers around zero such as between (0, 1), while r_1, r_2, v_1 and v_2 are four fixed constant values determined by the user such as (1, -1).

Algorithm 1:
Initialize population *(P[N])* with random solutions. Assign the values of r_1, r_2,v_1, v_2.
Evaluate *fitness* of the population, and keep the best solution in *Best.*
Divide population into 4 Groups: G_1, G_2, G_3 and G_4
Repeat
Calculate average points of the best solution (*Best)*, and store it in AV_{Best}
Case(1,2): for each cell in G_1 generate the new solution by using *reflection* and *visibility*, Equation (2,3), and calculate the *fitness.* Replace *current_fitness* and *Best_fitness* with the new *fitness* if the new *fitness* is better.
Case(3,4): for each cell in G_2 generate the new solution by using *reflection* and *visibility*, Equation (4,3), and calculate the *fitness.* Replace *current_fitness* and *Best_fitness* with the new *fitness* if the new *fitness* is better.
Case(5): for each cell in G_3 generate the new solution using *reflection* and *visibility*, Equation (5,6), and calculate the *fitness.* Replace *current_fitness* and *Best_fitness* with the new *fitness* if the new *fitness* is better.
Case(6): for each cell in G_4 generate a random solution and calculate the *fitness.* Replace *current_fitness* and *Best_fitness* with the new *fitness* if the new *fitness* is better.
Until stopping criteria
Return Best solution

2.3. Decision Tree

DT is one of the most well-known subfield of machine learning within the larger field of artificial intelligence. DTs are used widely by many researchers for classification problems. In addition, they are also assisting in uncovering features of data that were previously unrecognizable to the eye. therefore, they can be used successfully in data mining applications such as in (Lior Rokach, 2007). DTs are also work effectively in many different domains such as typical business scenarios for airplane autopilots and medical diagnoses (Mihai Lintean, 2007; Nahla Ben Amor, 2004).
DT classifier can be described as a recursive partition of the samples domain. It composes of nodes that create a rooted tree. Each inner node divides the instance domain into two or more subdomains based on some discrete functions of the input feature values. Each tested case is considered as a single feature, such that the instance domain is divided base on the attribute's value (Yacine Bouzida, 2006). Samples are classified by directing them from the root node to a leaf nods, based on the result of the tests along the path. The classification task starts from the root of a tree. It is considered the characteristic that corresponds to the root, and it determines which branch the value of the given characteristic is corresponded to. After that, the node where the given branch appears is evaluated. These processes will be repeated for each node until reaching a leaf node. Minimizing the entropy or information gain are two classical approaches of attribute selection are considered with the most famous algorithms such as ID3 and C4.5 (Quinlan, 1993).

3. PROPOSED DIDS DESIGN

The proposed system consists of several components which are: Collector Mobile Agent (CMA), Rule and Feature Generator Agent (RFGA), Controller Agent (CA), Action Mobile Agent (AMA), User Interface Agent (UIA), and several Nods (local networks) each with Sniffer Agent (SA) which works on the server as shown in Figure 2.
The work of the proposed system is described as follows: each node contains an agent called Sniffer Agent (SA). The goal of the SA is to collect connection information of incoming Internet packets and save them on a file, while CMA will migrate through the network moving from one node into another and gathers the information that is previously stored by SA. When CMA reaches the RFGA agent, it will pass its information to it

and the RFGA agent will use this information to generate and mine a subset of features with their corresponding rules. CFA is used to produce a subset of features while DT is considered as a measure on the produced features and generates a set of corresponding rules, both the generated rules and the generated features will be passed to CA. CA can then decide whether these features are a type of attacks or they constitute a normal behavior. The decision is based on the generated rules that are provided by RFGA agent. If controller finds out that there is an attack on one of the nodes, it then will take an action through the AMA agent such as shutting down the computer, or alerting the user through the UIA agent.

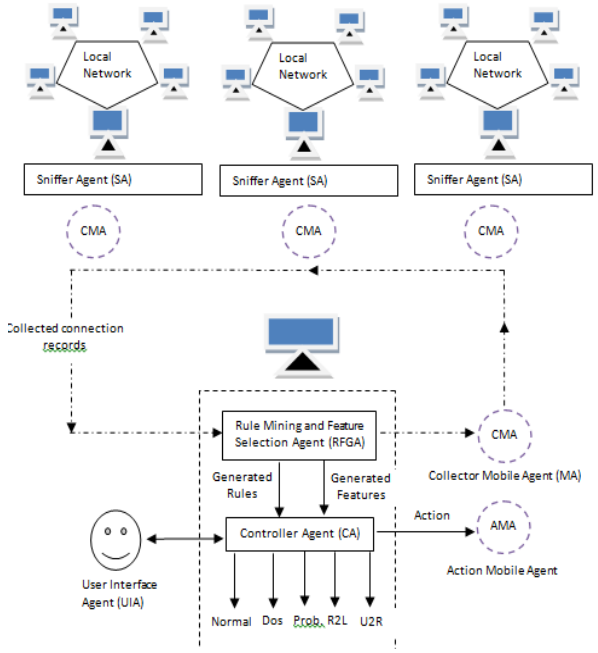

Figure 2. Proposed design of DIDS base MA

3.1. Rule and Feature Generator Agent

The general principle of the proposed agent using the combination of CFA and DT is shown in Figure 3. RFGA generates a subset of features by using CFA which are used to build a decision tree.

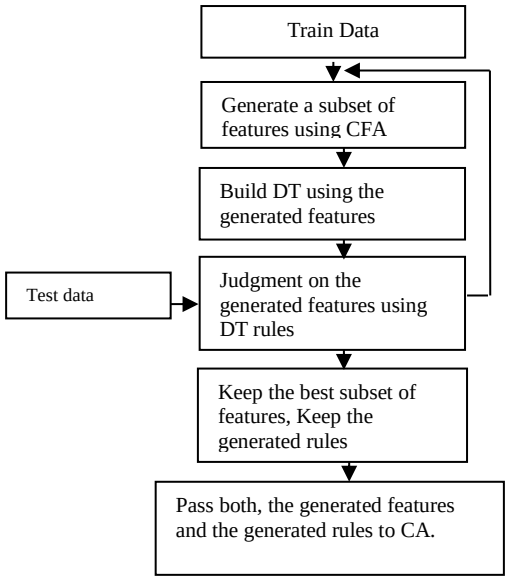

Figure 3. Flowchart of the combination of CFA and DT

The generated tree will be used as a judgment on the selected features by using the test dataset. These two steps will be repeated until the best subset of features is found. After that, both the generated features and the generated rule will be passed to CA.

3.2. Hierarchy of the Proposed System

Hierarchy of the proposed DIDS consists of five levels: in the first level, SA will gather connection information of the incoming packets from the network traffic and save them on a file, and this task will occur at each node in the network. In the second level, CMA will migrate through the networks and collect information that is stored by SA from each node by moving from a node into another until it reaches the RFGA and then it will pass its information about the visited nodes to the RFGA. In level 3, RFGA provides a subset of features with their corresponding rules which are produced by using the combination of the CFA and DT. The features subset and their corresponding rules are provided to be passed to CA. At level 4, the controller decides whether this information is a normal connection or an attack. The decision depends on several rules that are generated previously by DT. CA classifies each subset to one of the five classes resident in the KDD Cup 99 dataset: Normal, DoS, Probing, R2L, and U2R (Sandhya Peddabachigari, 2007). If CA agent finds that there is an attack, it will take an action through the AMA and the UIA in level 5. The hierarchy of the proposed DIDS is illustrated in Figure 4

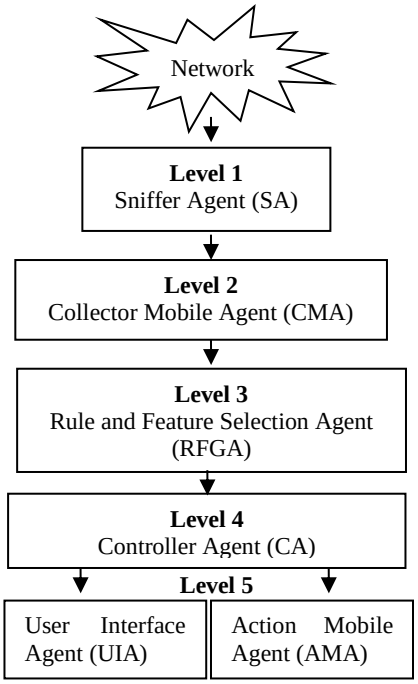

Figure 4. Hierarchy of the proposed DIDS

4. EVALUATION CRITERIA

For the evaluation process, the most known criteria used for KDD Cup 99 is the Cost Per Test (CPT) which is calculated by a confusion matrix and a given cost matrix (Elkan, 2000). A confusion matrix shown in Table 1, each column in this table is corresponding to the predicted class, while rows of this matrix are corresponding to the actual classes. The values of $CM(i, j)$, are the number of false classified samples that are actually belonging to class i, although falsely determined as a class j. The diagonal of this matrix $CM(i, i)$ is the number of truly detected samples.

Table 1. Confusion matrix

	Classifier-determined Positive label	Classifier-determined negative label
True positive label	a	b
True negative label	c	d

a = true positive, b = false negatives, c = false positive, d = true negative
- Accuracy = PSP = (a + d) / (a + b + c + d)
- Correct = a / (a + c)
- Recall = Detection Rate of class (a) = a / (a + b)

Cost matrix can be similarly defined, as well, and $C(i, j)$ is the cost penalty when a sample is falsely classified as class j which is actually belonging to class i. There are a standard values of Cost matrix shown in Table 2, these values are specially designed for the KDD Cup 99 (Eesa, 2011).

Table 2. Cost matrix

	Normal	Probing	DoS	U2R	R2L
Normal	**0**	1	2	2	2
Probing	1	**0**	2	2	2
DoS	2	1	**0**	2	2
U2R	3	2	2	**0**	2
R2L	4	2	2	2	**0**

CPT is calculated by using the following formula:

$$CPT = \frac{1}{N}\sum_{i=1}^{m}\sum_{j=1}^{m} CM(i, j) * C(i, j) \qquad (7)$$

Where CM is the confusion matrix and C is the cost matrix, while N and m are the total number of test samples and the number of the classes, respectively. The Accuracy Rate (AR) is based on the Percentage of Successful Prediction (PSP) and it can be calculated as follows:

$$AR = PSP = \frac{No.\ of\ correctly\ classified\ instance}{Total\ No.\ of\ instance\ in\ the\ test\ dataset} * 100\% \quad (8)$$

The result with high PSP and low CPT show better classification for the proposed system.

In addition to PSP and CPT, Detection Rate (DR) and Correct as a measurement were also used. DR is the ratio of the number of correctly classified samples as an attack to the total number of this attack in the test dataset. Correct criterion is the ratio of the number of correct classified instance to the total number of correct and incorrect instances and they can be calculated as follows:

$$DR = \frac{no.\ of\ \ correctly\ classified\ \ instances\ as\ an\ attack}{total\ no.\ of\ this\ attack\ in\ test\ \ dataset} * 100 \quad (9)$$

$$Correct = \frac{no.\ of\ \ correctly\ classified\ \ instances\ as\ class\ (i)}{total\ no.of\ \ correct\ and\ incorrect\ \ instances\ classified\ \ as\ class\ (i)} * 100 \quad (10)$$

Where $i = 1, 2, .., m$. m is the number of classes considered in KDD Cup 99 dataset. In this experiment, the DR, PSP, CPT and Correct measures are used to rank the different results.

5. EXPERIMENTS AND RESULTS

The proposed system was implemented under Visual Studio.NET 2010 environment using Visual C# language. The simulation have been carried out using laptop Dual-Core CPU 2.20 GHz, 2 GB RAM. Loopback address was used to simulate the network with different port addresses. The proposed system was tested by using three virtual hosts. One of the hosts represents the controller while others represent the nodes to be visited by the CMA agent. Complete 10%KDD Cup 99 train and test dataset which is given in Table 3 was used to test the proposed system. The subset of features that is used in this paper consists of only five features which are $\{f_3, f_{26}, f_{29}, f_{34}, f_{35}\}$. These features are produced after many experiments were implemented on the train dataset. To simulate the gathered instances from different SA in distributed manner, the test data is partitioned into two data sets equally.

Tables 4 and 5 are the confusion matrix associated with the DR, PSP, CPT and Correct produced by using the five features and the complete set of 41 features considered in 10%KDD Cup 99, respectively. From Tables 4 and 5, it can be easily seen that although five features are used, it performed better values of PSP and CPT when compared with the produced results using the complete 41 features. With DR, there is a slide difference between the two experiments for Normal and DoS attacks, although for U2R the use of 41 features led to a better result than the use of only five features. For R2L and Probing attacks, the use of five features give much better results than the results produced when using the complete set of 41 features. With Correct measure, the result of using five features is better than the result of using 41 features for all cases except for the U2R attack where the use of 41 features yields better performance.

Table 3. 10%KDD Cup 99 train and test datasets

Normal(97,277; 60,593)	
Probing (4, 107; 4, 166)	**DoS(391, 458; 229, 853)**
ipsweep(1, 247; 306), mscan(0; 1, 053), nmap(231; 84), portsweep(1, 040; 364), saint(0; 736), satan(1, 589; 1, 633).	apache2(0; 794), back(2, 203; 1.098), land(21; 9), mailbomb(0;5, 000), neptune(107, 201; 58, 001), pod(264;87), processtable(0; 759), smurf(280, 790; 164, 091), teardrop(979; 12), udpstorm(0; 2).
U2R(52; 228)	**R2L(1, 126; 16, 189)**
buffer overflow(30, 22), httptunnel(0; 158), loadmodule(9; 2), perl(3; 2),perl(3; 2), ps(0; 16), rootkit(10;13), sqlattack(0; 2), xterm(0; 13).	ftp write(8; 3), imap(12; 1), multihop(7; 18), named(0; 17), phf(4;2), sendmail(0;17), snmpgetattack(0; 7, 741), guess passwd(53; 4, 367), snmpguess(0; 2, 406), spy(2; 0), warezclient(1, 020; 0), warezmaster(20;1, 602), worm(0;2),xlock(0; 9), xsnoop(0; 4).
Total Train data set = 494020 **Total Test data set = 311028**	

Table 4. Confusion matrix related to the DR, PSP, CPT and Correct using subset of five features

Predicted Actual	Normal	Prob	DoS	U2R	R2L	%DR
Normal (60,591)	**60036**	167	241	0	147	99.1
Probing (4,166)	362	**298 9**	712	1	102	71.75
DoS (229,853)	7274	100	**222147**	4	328	96.65
U2R (228)	206	0	0	**0**	22	0
R2L (16,189)	14592	0	74	0	**1523**	9.41
Correct	72.8	91.8	99,5	0	71.8	
5 features		***PSP* = 92.177%**			***CPT* = 0.2489**	

Table 5. Confusion matrix related to the DR, PSP, CPT and Correct using complete 41 features

Predicted Actual	Normal	Prob	DoS	U2R	R2L	%DR
Normal (60,591)	**60223**	243	109	9	5	99.4
Probing (4,166)	601	**2862**	700	0	3	68.7
DoS (229,853)	7124	300	**222431**	0	0	96.77
U2R (228)	191	0	0	**36**	1	15.8
R2L (16,189)	15646	13	514	11	**5**	0.03
Correct	71.88	83.73	99.41	64.28	35.71	
41 features		***PSP* = 91.811%**			***CPT* = 0.2613**	

6. CONCLUSION

The combination model of CFA and DT as an agent for feature selection in DIDS is investigated and tested on the benchmark KDD Cup 99 dataset. In addition, the migration of the CMA through the networks was simulated and tested successfully. After many experiments the best five features produced with CFA was {f_3, f_{26}, f_{29}, f_{34}, f_{35}}. The performance of these five features was compared with the performance of the complete 41 features in the KDD Cup 99 dataset. The obtained results show that with the only five features, the proposed system performs better than the complete 41 features.

The investigation of using CFA as a rule generator for IDS can be suggested as a future work. Moreover, the use of other techniques such as support vector machines, neural networks, clustering methods instead of using DT remains an open issue. Comparisons of feature selection techniques will also provide clues for constructing more effective models for intrusion detection.

REFERENCES

Adel Sabry Eesa, A. M. A. B., Zeynep Orman. (2013). Cuttlefish Algorithm – A Novel Bio-Inspired Optimization Algorithm. *International Journal of Scientific & Engineering Research, 4*(9).

Adel Sabry Eesa, Z. O., Adnan Mohsin Abdulazeez Brifcani. (2015). A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems. *Expert Systems with Applications, 42*, 2670–2679.

The Agent-Oriented Software Engineering Handbook. (2004). In M.-P. G. Federico Bergenti, Franco Zambonelli (Ed.), *Methodologies and Software Engineering for Agent Systems* (Vol. 11): Springer US.

Center, I. A. T. A. (2009). *Information Assurance Tools Report: Intrusion Detection Systems*: Information Assurance Technology Analysis Center.

Chi-Ho Tsang, S. K., Hanli Wang. (2007). Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection. *Pattern Recognition, 40*(9), 2373–2391.

Dalila Boughaci, H. D., Ahmed Bendib, Youcef Bouznit (2006, 25-27 May 2006). *Distributed Intrusion Detection Framework based on Autonomous and Mobile Agents.* Paper presented at the 2006 International Conference on Dependability of Computer Systems, Szklarska Poreba.

Donald G. Marks, P. M., Michael Stinson. (2004). Optimizing the Scalability of Network Intrusion Detection Systems Using Mobile Agents. *Journal of Network and Systems Management 12*(1), 95-110

E., M. (2005). A New Mobile Agent-Based Intrusion Detection System Using Distributed Sensors. *American University of Beirut*.

Eesa, A. S. (2011). Intrusion Detection System Based on Decision Tree and Clustered Continuous Inputs. *Raf. J. of Comp. & Math's, 8*(1).

Elkan, C. (2000). Results of the KDD'99 classifier learning. *SIGKDD Explor. Newsl., 1*(2), 63-64.

Hai Thanh Nguyen , K. F., Slobodan Petrovic. (2010). *Towards a Generic Feature-Selection Measure for Intrusion Detection*. Paper presented at the Pattern Recognition (ICPR), 2010 20th International Conference on.

Imen Brahmi, S. B. Y., Pascal Poncelet. (2010). MAD-IDS: Novel Intrusion Detection System Using Mobile Agents and Data Mining Approaches *Intelligence and Security Informatics* (pp. 73-76): Springer Berlin Heidelberg.

Jean-Louis Lassez , R. R., Stephen Sheel , Srinivas Mukkamala. (2008). *Signature based intrusion detection using latent semantic analysis*. Paper presented at the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence).

Lior Rokach, O. M. (2007). *Data Mining With Decision Trees, Theory and Applications*: World Scientific.

Manmeet S, S. S. S. (2007). *Distributed Intrusion Detection System using Mobile Agents* Paper presented at the National Conference on Challenges & Opportunities in Information Technology (COIT).

Mihai Lintean, V. R. (2007). *Naive Bayes And Decision Trees For Function Tagging*. Paper presented at the of the International Conference of the FLAIRS-2007, Key West, FL.

Mohanabharathi R, M. T. K., Dr.S.Karthi. (2012). Feature Selection for Wireless Intrusion Detection System Using Filter and Wrapper Model. *International Journal of Modern Engineering Research (IJMER), 2*(4), 1552-1556.

N. Pratik Neelakantan , C. N. M. T. (2011). Role of Feature Selection in Intrusion Detection Systems for 802.11 Networks. *International Journal of Smart Sensors and Ad Hoc Networks (IJSSAN) 1*(1), 98-101.

Nahla Ben Amor, S. B., Zied Elouedi. (2004). *Qualitative classification with possibilistic decision trees.* Paper presented at the the International Conference on Information Processing of Uncertainty in Knowledge Based Systems IPMU'2004,, Perugia, Italy.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*: Morgan Kaufmann Publishers.

R. Sasikumar, D. M. (2012). Dynamic Distributed Intrusion Detection System Based on Mobile Agents with Fault Tolerance. *Journal of Computer Science, 8*(7), 1092-1098.

Rupali Datti, S. L. (2012). Performance Comparison of Features Reduction Techniques for Intrusion Detection System *International Journal of Computer Science And Technology, 3*(1).

Saidat Adebukola Onashoga, A. D. A., Adesina Simon Sodiya (2009). *A Strategic Review of Existing Mobile Agent-Based Intrusion Detection Systems*. Retrieved from http://iisit.org/Vol6/IISITv6p669-682Onashoga623.pdf

Sandhya Peddabachigari, A. A., Crina Grosan, Johnson Thomas. (2007). Modeling intrusion detection system using hybrid intelligent systems. *Journal of Network and Computer Applications, Elsevier, 30*(1), 14–132.

Shih-Wei Lin, K.-C. Y., Chou-Yuan Lee, Zne-Jung Lee. (2012). An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection. *Applied Soft Computing, 12*(10), 3285–3290.

Steven R. Snapp, J. B., Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che-lin Ho, Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha, Tim Grance, Daniel M. Teal, Doug Mansur. (1991). *DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype*. Paper presented at the Snapp91dids.

V. Bolón-Canedo, N. S.-M., A. Alonso-Betanzos (2011). Feature selection and classification in multiple class datasets: An application to KDD Cup 99 dataset. *Expert Systems with Applications, 38*(5), 5947–5957.

Wang J., C.-j. W., Jun-yuan X., Shi-fu C. . (2006). Research on Agent-based Intrusion Detection Technique.Computer Science. *Computer Science, 33*(12).

Yacine Bouzida, F. C. (2006). Neural networks vs. decision trees for intrusion detection. *IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM), Tuebingen, Germany, 28*(29).