Genetic Algorithm Combined with Support Vector Machine for Building an Intrusion Detection System

Sriparna Saha
Department of CSE
IIT Patna
Patna, India
sriparna@iitp.ac.in

Ashok Singh Sairam
Department of CSE
IIT Patna
Patna, India
ashok@iitp.ac.in

Asif Ekbal
Department of CSE
IIT Patna
Patna, India
asif@iitp.ac.in

Amulya Yadav
Department of CSE
IIT Patna
amulya@iitp.ac.in

ABSTRACT

In this paper, we develop an intrusion detection system (IDS) based on machine learning. We employ genetic algorithm (GA) along with Support Vector Machine (SVM) for automatically determining the appropriate set of features. The idea is then developed into a fully functional IDS. Experiments of testing the IDS on the benchmark KDD CUP 99 datasets are presented. Results show encouraging performance that opens a avenue for further research.

Categories and Subject Descriptors

C.2 [**COMPUTER-COMMUNICATION NETWORKS**]: GeneralSecurity and protection; C.2.0 [**General**]: Security and protection

General Terms

Network Security

Keywords

Genetic Algorithms; Support Vector Machines; Feature Selection; Intrusion Detection System

1. INTRODUCTION

KDD 99 intrusion detection datasets, which are based on the DARPA 98 dataset, provides labeled data for researchers working in the field of intrusion detection and is the only labeled dataset publicly available. These datasets have been used by various researchers in intrusion detection competitions to study the utilization of machine learning for intrusion detection and reported detection rates up to 91% with false positive rates less than 1%. The seminal paper on the KDD CUP 99 dataset by Kayacik et al. [1] investigates the relevance of each feature in KDD 99 intrusion detection datasets(there are 41 in total). In the KDD CUP dataset, a total of 23 different attack types are present and the most discriminating features for each class are presented in [1]. However, the sets of discriminating features presented in [1] are not accurate enough. At first we carried out some preliminary experiments on their proposed approach for determining the most discriminating features and accordingly found the necessities of developing a systematic approach to select the features more efficiently. We present a new method of finding out the most important features of the dataset for each attack type and finally present a machine learning IDS which needs to be trained once on the training data and can then successfully classify incoming packets as either normal or attack packets. The organization of the paper is as follows:

Section 2 describes other machine learning approaches used in making an IDS. Section 3 describes the method used in [1] to find out the most discriminating features of the datasets. Section 4 provides a brief overview of genetic algorithm (GA) and Support Vector Machine (SVM). In section 5, we present our method of finding out the most discriminating features of the datasets. Section 6 provides the results that we obtained. Section 7 finally throws some light on our IDS. Section 8 is the concluding section and future work is discussed in that.

ICACCI'12, August 3-5, 2012, Chennai, T Nadu, India.

2. RELATED WORK

Over the past few years, researchers have been divided on two different perspectives of solving the problem of intrusion detection.

- Converting the problem into that of a classification problem on network state (and not on individual packets or other units) by modeling normal and attack traffic and classifying the current state of the network as good or bad, thereby detecting attacks when they happen. Classical machine learning algorithms are used to solve the problem
- Cryptographic solutions like client puzzles which discourage attack attempts.

Various methods have been proposed to accomplish the above objective. They include statistical approaches, like [2], which proposes a Chi-Square-Test on the entropy values of the packet headers. In [3] authors discussed the effects of multivariate correlation analysis on the DDoS detection and presented an example of SYN flooding. The covariance matrix is used to present the relationship between each pair of network feature and identifies attacks in [4]. In [5] the CUSUM algorithm is used to detect the change in the amount of packets to destination. Different other techniques taken from pattern analysis and machine learning have also been proposed in literature. For example, Markov Models [11] consider application layer DDoS attacks and use hidden semi-markov models to describe browsing behavior of users for anomaly detection at the application layer. Hybrid modeling techniques such as [12] also provide interesting results. Hidden Markov Model based DoS attack solutions have also been proposed in [13]. A taxonomy of DDoS attacks and defence mechanism has been documented in [14]. Recent works [15] have discussed use of bayesian classifiers towards intrusion detection in general, which includes DoS attacks. A clustering algorithm is used to solve the same problem in [16] to build an IDS using KDD data. A combination of Principal Component Analysis (PCA) and SVM is used in [17] to build an IDS.

In these works, DoS attacks have been classified to fall into the broad category of Intrusion attacks, and so solutions have been oriented more towards formal intrusions, including root escalation, scripting attacks etc. A major factor that is often missed by classifying DoS attacks into intrusion attacks is the enormous volume that DoS detection solutions have to handle in comparison with intrusion attacks. This fact straightaway differentiates DoS attacks from other types of intrusion, and renders learning models such as SVMs, HMMs, ANNs impractical when it comes to real time attack detection. The detection mechanism is expected to be ultra light weight to support speeds in the order of at least a few hundred Mbps. This is also well complimented by the fact that the very nature of DoS attacks (the system can absorb some attack traffic unlike intrusions) makes it detectable using simpler notions than the ones which use input parameters of models described above. Another practical drawback of supervised/unsupervised learning models for DoS detection is the accuracy of (supposedly normal) traffic supplied to learning. In practical user sites, it is a very difficult proposition to be fully confident that the input to learning is absolutely normal. The system may cause false alarms if traces of the training traffic contained abnormalities. For making practical systems out of research outcomes, additional work has to be done to eliminate considering such abnormalities which may be present in traffic supplied to learning.

Another common problem to the entire research community on DoS attacks is the lack of training data. The DoS research community depends extensively on standard datasets (KDD dataset [18])for the purpose of learning and analysis, which are better suited to analysis of intrusion attacks.

So a practical system using any machine learning algorithm to detect DoS attacks should be carefully engineered to be a) Light weight; b) Accommodative of practical difficulties in learning. Different systems for DoS detection and mitigation have different approaches to the problem based on their position in the network. The system could be placed either near the target, or the source, or anywhere at an intermediate point in the network[9].

The problem in [16] is that it provides a very low accuracy of classification which is unacceptable by today's standards. However, it is one of the simplest approaches to building an IDS using machine learning. A major drawback of [17] is that by using PCA, we are distorting the original data and thus, this approach will not be scalable to any other dataset other than KDD, or for that matter, any dataset on which we run our algorithm. Even if we want to add a new feature in the KDD dataset to analyze its effect on the classification, we would be left handicapped and would have to run the entire algorithm again.

3. ORIGINAL METHOD

Kayacik et al[1] in their work, use an approach based on information gain. Based on the entropy of a feature, information gain measures the relevance of a given feature, in other words its role in determining the class label. If the feature is relevant, in other words highly useful for an accurate determination, calculated entropies will be close to 0 and the information gain will be close to 1. Since information gain is calculated for discrete features, continuous features are discretized with the emphasis of providing sufficient discrete values for detection.

Information gain is calculated for class labels by employing a binary discrimination for each class. That is, for each class, a dataset instance is considered in-class, if it has the same label; out-class, if it has a different label. Feature relevance is expressed in terms of information gain, which gets higher as the feature gets more discriminative. In order to get feature relevance measure for all classes in training set, information gain is calculated on binary classification, for each feature resulting in a separate information gain per class.

4. OVERVIEW OF TECHNIQUES USED

4.1 Genetic algorithm

A genetic algorithm [9] is basically a search method for finding the optimal or near-optimal solutions in a reasonable amount of time. Note that the solution returned by a genetic algorithm is almost never optimal, it is close to optimality. However, searching through all possible solutions for the optimal one would take an unacceptable amount of time as opposed to a genetic algorithm.

GA's begin with a set of k randomly generated states, called **population**. Each state, or individual, is represented as a string over a finite alpabet-more commonly, a string of 0's and 1's. This representation of an individual is called a chromosome.

For the production of next generation of states, each state is rated by the fitness function. A fitness function should return higher values for better states. The genetic algorithm then proceeds in several steps:

- **Selection**

  In a common form of selection, known as fitness proportional selection, each chromosome's likelihood of being selected as a good one is proportional to its fitness value.The alteration step in the genetic algorithm refines the good solution from the current generation to produce the next generation of candidate solutions. It is carried out by performing crossover and mutation.

- **Crossover**

  Crossover may be regarded as artificial mating in which chromosomes from two individuals are combined to create the chromosome for the next generation. This is done by splicing two chromosomes from two different solutions at a crossover point and swapping the spliced parts. The idea is that some genes with good characteristics from one chromosome may as a result combine with some good genes in the other chromosome to create a better solution represented by the new chromosome.

- **Mutation**

  Mutation is a random adjustment in the genetic composition. It is useful for introducing new characteristics in a population; something not achieved through crossover alone. Crossover only rearranges existing characteristics to give new combinations. For example, if the first bit in every chromosome of a generation happens to be a 1, any new chromosome created through crossover will also have 1 as the first bit. The mutation operation changes the current value of a gene to a different one. For bit string chromosome this change amounts to flipping a 0 bit to a 1 or vice versa. And each bit has a mutation probability of being flipped.

A genetic algorithm pseudocode is presented below:

Algorithm 1 Genetic Algorithm

Require: population, a set of individuals
FITNESS, a function that measures fitness of an individual
Ensure: near optimal individual
while no individual is fit enough **do**
newpopulation ← *emptyset*
for $i \leftarrow 1$ *to* $SIZE(population)$ **do**
$x \leftarrow SELECT(population, FITNESS)$
$y \leftarrow SELECT(population, FITNESS)$
$child \leftarrow REPRODUCE(x, y)$
if small random probability **then**
$child \leftarrow MUTATE(child)$
end if
add child to newpopulation
end for
population ← *newpopulation*
end while
return the best individual in the population, according to FITNESS

4.2 Support Vector Machine

Support Vector Machine [8] is a new technique for data classification which is proposed by Vapnik and based on Statistical Learning Theory. SVM is a common learning method suiting for small size example set. It finds a global minimum of the actual risk upper bound using structural risk minimization and avoids complex calculation in high dimensional space by kernel function. In essence, SVM trains itself on a labeled training file and builds a model and then classifies test data according to the built model.

Suppose, we have a set of training data for a two-class problem: $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where $\mathbf{x}_i \in R^D$ is a feature vector of the i-th sample in the training data and $y \in \{+1, -1\}$ is the class to which $\mathbf{x}_i$ belongs. In their basic form, a SVM learns a linear hyperplane that separates the set of positive examples from the set of negative examples with *maximal margin* (the margin is defined as the distance of the hyperplane to the nearest of the positive and negative examples). In basic SVMs framework, we try to separate the positive and negative examples by the hyperplane written as:

$$(\mathbf{w}.\mathbf{x}) + b = 0 \qquad \mathbf{w} \in \mathbf{R}^n, b \in \mathbf{R}.$$

SVMs find the "optimal" hyperplane (optimal parameter $\overline{w}, b$) which separates the training data into two classes precisely.

The linear separator is defined by two elements: a weight vector $\mathbf{w}$ (with one component for each feature), and a bias b which stands for the distance of the hyperplane to the origin. The classification rule of a SVM is:

$$sgn(f(\mathbf{x}, \mathbf{w}, b)) \tag{1}$$

$$f(\mathbf{x}, \mathbf{w}, b) = <\mathbf{w}.\mathbf{x}> + b \tag{2}$$

being $\mathbf{x}$ the example to be classified. In the linearly separable case, learning the maximal margin hyperplane $(\mathbf{w}, b)$ can be stated as a convex quadratic optimization problem with a unique solution: *minimize* $||\mathbf{w}||$, *subject to the constraints* (one for each training example):

$$y_i(<\mathbf{w}.x_i> + b) \geq 1 \tag{3}$$

The SVM model has an equivalent dual formulation, characterized by a weight vector α and a bias b. In this case, α contains one weight for each training vector, indicating the importance of this vector in the solution. Vectors with non null weights are called *support vectors*. The dual classification rule is:

$$f(\mathbf{x}, \alpha, b) = \sum_{i=1}^{N} y_i \alpha_i <\mathbf{x}_i.\mathbf{x}> + b \tag{4}$$

The α vector can be calculated also as a quadratic optimization problem. Given the optimal α^* vector of the dual quadratic optimization problem, the weight vector $\mathbf{w}^*$ that realizes the maximal margin hyperplane is calculated as:

$$\mathbf{w}^* = \sum_{i=1}^{N} y_i \alpha_i^* \mathbf{x}_i \tag{5}$$

The b^* has also a simple expression in terms of $\mathbf{w}^*$ and the training examples $(\mathbf{x}_i, y_i)_{i=1}^N$.

The advantage of the dual formulation is that efficient learning of non-linear SVM separators, by introducing *kernel*

functions. Technically, a *kernel function* calculates a dot product between two vectors that have been (non linearly) mapped into a high dimensional feature space. Since there is no need to perform this mapping explicitly, the training is still feasible although the dimension of the real feature space can be very high or even infinite.

By simply substituting every dot product of $\mathbf{x}_i$ and $\mathbf{x}_j$ in dual form with any *kernel function* $K(\mathbf{x}_i, \mathbf{x}_j)$, SVMs can handle non-linear hypotheses. Among the many kinds of *kernel functions* available, we will focus on the d-th *polynomial kernel*:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i.\mathbf{x}_j + 1)^d$$

Use of d-th polynomial kernel function allows us to build an optimal separating hyperplane which takes into account all combination of features up to d.

Support Vector Machines have advantage over conventional statistical learning algorithms from the following two aspects:

1. SVMs have high generalization performance independent of dimension of feature vectors.
2. SVMs can carry out their learning with all combinations of given features without increasing computational complexity by introducing the *Kernel function*.

5. FEATURE SELECTION

The performance of any classification technique depends on the features of training and test data sets. Feature selection [10, 6], also known as variable selection, attribute selection or variable subset selection, is the technique, commonly used in machine learning, of selecting a subset of relevant features for building robust learning models. By removing most irrelevant and redundant features from the data, feature selection helps to improve the performance of a classifier. Feature selection [7] is important for the following reasons (i). to alleviate the effect of the curse of dimensionality, (ii). to enhance generalization capability, (iii). to speed up learning process and (iv). to improve model interpretability. Appropriate feature selection is very important for any classifier. Use of more features may create the problem of loss of generalization whereas use of less features sometimes causes degradation in classification quality. Feature selection also helps people to acquire better understanding about their data by telling them which are the important features and how they are related with each other.

In general, feature selection problem is formulated under the single objective optimization. It is stated as follows: Given a set of features Ω and a classification quality measure P, determine the feature subset F^* such that:

$$P(F^*) = \max_{F \in \Omega} P(F)$$

In general the search space for this type of problems is 2^d, where d is the total number of possible features. Thus, exhaustive search strategies can not be applied in this case. Some heuristics based techniques like GA [9] can be used to search for the appropriate feature combination.

6. OUR METHOD

In Section 3, we outlined the approach employed by Kayacik et. al. in finding the most discriminating features for classification for each attack type. For each attack type, they found out the information gain per feature. The greater the information gain, the most discriminating that feature is. Thus, they sort features based on the information gain and keep all features above a threshold limit in their discriminating set. However, this approach has a serious flaw. Kayacik et. al. only consider features in their singularity. However, it is quite possible that the combined effect of 2 or more features would have a more discriminatory effect on classification than features in singular. We wanted to exploit this weakness in their paper and see if we can improve the classification accuracies.

We had several tools at our disposal such as Principal Component Analysis(PCA) and Genetic Algorithm(GA) in order to find out the most relevant features of the KDD CUP 99 data. However, using PCA involved the overhead of losing the original data in order to get our data into lesser dimensions. We would have to transform our data into new, less dimensional data while using PCA. We did not want that to happen, because in the future, we might want to add new features in the KDD dataset and see whether the new feature comes up as a discriminatory feature. So, modifying the data was out of the question. Hence, we opted to use a genetic algorithm in order to find out the most discriminating subset of features. In order to classify the data, support vector machine based classification technique is used. Thus we have developed a new technique based on GA coupled with SVM to identify relevant features for any intrusion detection system.

6.1 Proposed GA Based Feature Selection Technique

In this section, we describe our proposed single objective optimization (SOO) based feature selection approach. It identifies the relevant sets of feature for SVM based classifier. The proposed approach is based on GA [9] that closely follows those of the steps as shown in Figure 1. The pseudo code of the proposed algorithm is presented in Figure 2.

6.2 Chromosome Representation and Population Initialization

If the total number of features is F, then the length of the chromosome is F. As an example, the encoding of a particular chromosome is represented in Figure 1. Here, $F = 12$ (i.e., total 12 different features are available). The chromosome represents the use of 7 features, i.e., first, third, fourth, seventh, tenth, eleventh and twelfth for constructing the particular SVM based classifier. The entries of each chromosome are randomly initialized to either 0 or 1. Here, if the i^{th} position of a chromosome is 0 then it represents that i^{th} feature does not participate in constructing the classifier. Else, if it is 1 then the i^{th} feature participates in constructing the classifier.

If the population size is P then all the P number of chromosomes of this population are initialized in the above way. The KDD CUP 99 dataset has 41 features in total. Thus the chromosome in our genetic algorithm is 41 bits long.

6.3 Fitness Computation

We use SVM for this purpose. For each subset considered,

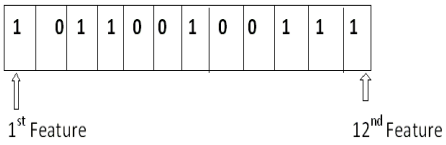

Figure 1: Chromosome representation for GA based feature selection

we take the 10 percent labeled KDD dataset and used that as the test file. The remaining 90% data is used as the training set. The following steps are performed for each chromosome.

1. Suppose, there are N number of features present in a particular chromosome (i.e., there are total N number of 1's in that chromosome).
2. Construct the SVM based classifier with only these N features.
3. The SVM based classifier is tested on the test data.
4. The overall error rate of this SVM based classifier is calculated.

The objective function corresponding to a particular chromosome is:
f = error_rate
The objective is to minimize this objective function.

6.4 Genetic Operators

Roulette wheel selection is used to implement the proportional selection strategy. We use the normal single point crossover [9]. As an example, let the two chromosomes be :
$P1$: 0 1 1 0 0 0 1 0
$P2$: 1 1 1 0 0 0 0 1
At first a crossover point has to be selected uniformly random between 1 to 8 (length of the chromosome) by generating some random number between 1 and 8. Let the crossover point, here, be 4. Then after crossover, the offsprings are:
$O1$: 0 1 1 0 0 0 0 1 (taking the first 4 positions from $P1$ and rest from $P2$)
$O2$: 1 1 1 0 0 0 1 0 (taking the first 4 positions from $P2$ and rest from $P1$)

Crossover probability is kept equal to 0.9.

Each chromosome undergoes mutation with a probability 0.2. Here, the value present at each position in a chromosome is flipped, i.e., if it initially contains 1 it will be replaced by 0; if it contains 0 it will be replaced by 1.

6.5 Termination Condition

In this approach, the processes of fitness computation, selection, crossover, and mutation are executed for a maximum number of generations. The best string seen up to the last generation provides the solution to the above classifier ensemble problem. Elitism is implemented at each generation by preserving the best string seen up to that generation in a location outside the population. Thus on termination, this location contains the best classifier ensemble.

7. RESULTS

Collection of results was completed in several phases. First of all, it was verified whether our proposed method would actually work ie. does our method actually reduce the error rate for classification by SVM.

Algorithm 2 Feature discriminator

Require: data set: KDD 99 dataset
numgen: number of generations to run the GA
POP_SIZE: size of the population
μ_m: probability of mutation
μ_c: probability of crossover
Ensure: most discriminating subset of features
initialize the chromosomes of the population randomly
while $numgen \geq 0$ **do**
for i=1 to POP_SIZE **do**
decode the feature subset encoded in chromosome i
$newKDDfile \leftarrow TRIM(KDDfile)$
$SVMTRAIN(newKDDFILE)$
$FITNESS(S) \leftarrow SVMTEST(S)$
end for
apply mutation, crossover, selection
select the best chromosome with respect to fitness
end while
return the best individual in the population, according to FITNESS

So first of all, we used all 41 features and used this KDD file to train SVM and then used a test file to find out the accuracy of classification. Then we fed the file into the genetic algorithm. If the error rate of classification of the best subset given by GA is lessser than the original error rate, then that would mean that our method is working well.

The results are reported in Table 2. This table shows that error rate after application of GA reduced by 9%. This supports the use of GA for feature selection of SVM based classifier.

Once the validity of the method was established, we went on to establish the most relevant features for each attack type. For that, we created a new training file for each attack type. For each attack type, say smurf, we would relabel the entire file in the following way: Each connection record, which was labelled as smurf, was labelled as 1. All other records were labelled as 2. Then this file would be used in our GA to find out the most relevant subset of features. For the purpose of comparison we have also created another sets of training and test files of KDD keeping only the features identified by [1] for a particular attack type. SVM is trained with the corresponding training file and tested with the test file. Error rates are calculated. These are compared with those obtained by our proposed method. Table 3 shows the comparison results. This table shows that for most of the data types except smurf attach type, our approach achieves a reasonable amount of performance improvements over approach in [1]. For example for normal data type while our proposed approach attained 2.14% error rate, approach proposed in [1] attained the error rate of 18.1447%. This is an improvement of more than 16%. Similarly for land attack type, there is an improvement of more than 99%.

8. THE IDS

In section 5, we detailed our approach towards getting the most discriminating features for each attack type. So, now

METHOD USED	ERROR RATE
Without GA	22%
With GA	13%

Figure 2: Table showing the effectiveness of GA

RESULTS OF PERFORMING EXPERIMENTS

ATTACK TYPE	Our Error rate	Our set of features	Their set of feat.	Their error
Normal	Min-2.14 %	1,3,4,5,9,10,12,13,14,18,20,22,23,24,25,30,34,35,39,40	1,6,12,15,16,17,18,19,31,32,37	18.1447%
SMURF	0.00000	5,8,9,13,14,15,,22,24,25,26,31,32,33,34,36,37,39,41	2, 3, 5, 23, 24, 27, 28, 36, 40, 41	0
NEPTUNE	0.053%	1,2,3,4,5,6,7,8,9,10,12,16,18,19,21,23,24,26,27,30,31,33,34,35,36,38,40,41	4, 25, 26, 29, 30, 33, 34, 35, 38, 39	4.227 %
LAND	0.004%	5,8,9,13,14,15,22,24,25,26,31,32,33,34,36,37,39,41	7	99.9958%
Teardrop	0.242%	5,6,7,8,11,12,13,14,16,20,25,26,28,30,37,38,40,41	8	99.7579%
back	0.446%	5,8,9,13,14,15,22,24,25,26,31,32,33,34,36,37,39,41	10,13	99.7334%
ftp_write	0.001%	5,8,9,13,14,15,22,24,25,26,31,32,33,34,36,37,39,41	9	99.9988%
Guess_passwd	0.011%	5,8,9,13,14,15,22,24,25,26,31,32,33,34,36,37,39,41	11	99.9891%

Figure 3: Comparison results of our system and the approach proposed in [1]

say that we know the most discriminating subset using our method. An IDS should be able to detect malicious incoming packets, in real time. We have a predefined dictionary of attack types(in KDD's case, we have 23) and by using the GA and SVM approach, we know the most discriminating features for each attack type. So, an incoming packet could be run under SVM to classify it as either normal or as one of the attack types. Better still, if finding out the kind of attack is not important to us and we just want the IDS to serve as an alarm for attack packets,we could change our binary discrimination strategy accordingly. If a data instance is labelled as some kind of attack, then it is in class; otherwise it is out-class. This way, we would be getting a different subset of most discriminating features which would be best at discriminating attack packets from normal traffic packets. The best part about this IDS is that all the memory intensive computation for finding out the most discriminating subset for a particular class is done offline and is done only once. So, there is a constant preprocessing overhead associated with the IDS. And to top it off, it is very accurate with a very low false positive rate.

9. FUTURE WORK

In this paper, the development of an IDS using machine learning techniques was described. The IDS was tested on KDD CUP dataset and was found to more accurate than any other existing IDS using machine learning. In future, we would like to improve our IDS using multi objective optimization GA. Right now, we are using single objective GA and we expect that by using multi objective GA, we will improve the rate of error finding in our IDS. However, some thought will need to be put into what other parameters can be used as the other objective functions.

10. REFERENCES

[1] H G Kayacik, A N Zincir-Heywood, M I Heywood, *Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets*, Proceedings of the Third Annual Conference on Privacy Security and Trust PST-2005 (2005) Publisher: Citeseer, Pages: 3-8.

[2] Tirthankar Ghosh, Niki Pissinou and Kia Makki *Statistical approaches to ddos attack detection and response,* , in Proc. DARPA Information Survivability Conference and Exposition, 2003, vol. 1, Apr. 2003, pp. 303-314.

[3] S. Jin and D. Yeung *A covariance analysis model for ddos attack detection,*, 2004 IEEE International Conference, China, Jun. 2004, pp. 1882-1886.

[4] S. Jin and D. Yeung *Ddos detection based on feature space mod- eling*,in Proc. of 2004 International Conference on Machine Learning and Cybernetics, vol. 7. IEEE Press, Aug. 2004, pp. 4210-4215.

[5] Z. Zhou, D. Xie, and W. Xiong, *A novel distributed detection scheme against ddos attack*,Journal of Networks(JNW), vol. 4, no. 9, pp. 921-928, Nov 2009.

[6] Liu, H., Motoda, H.: *Feature Selection for Knowledge Discovery and Data Mining*, Kluwer Academic Publishers, Norwell, MA, USA, 1998, ISBN 079238198X.

[7] Isabelle Guyon and Andr Elisseeff: *An introduction to variable and feature selection.* J. Mach. Learn. Res. 3 (March 2003), 1157-1182.

[8] C.Cortes and V.Vapnik. *Support vector networks.* Machine Learning, 20:273–297, 1995.

[9] Goldberg, D. E.: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, New York, 1989.

[10] Liu, H., Yu, L.: Toward Integrating Feature Selection Algorithms for Classification and Clustering, *IEEE Trans. on Knowl. and Data Eng.*, **17**(4), 2005, 491–502, ISSN 1041-4347.

[11] Y. Xie and S.-Z. Yu, *A novel model for detecting application layer ddos attacks,*In Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences - Volume 2 (IMSCCS'06) - Volume 02 (IMSCCS '06), Vol. 2. IEEE Computer Society, Washington, DC, USA, 56-63. DOI=10.1109/IMSCCS.2006.159 http://dx.doi.org/10.1109/IMSCCS.2006.159.

[12] T. Shon, Y. Kim, C. Lee, and J. Moon, *A machine learning framework for network anomaly detection using svm and ga*,in Information Assurance Workshop, 2005. IAW 2005. Proceedings from the Sixth Annual IEEE SMC, Nagoya, Japan, Jun. 2005, pp. 176-183.

[13] J. Kang, Y. Zhang, and J. bin Ju, *Detecting ddos attacks based on multi- stream fused hmm in source-end network*,in International Conference on Cryptography and Network Security, 2006. in International Conference on Cryptography and Network Security, 2006.

[14] J. Mirkovic and P. Reiher, *A taxonomy of ddos attack and ddos defense mechanisms*,SIGCOMM Comput. Commun. Rev., vol. 34, no. 2, pp. 39-53, 2004.

[15] D. M. Farid, N. Harbi, and M. Z. Rahman, *Combining naive bayes and decision tree for adaptive intrusion detection*,CoRR, vol. abs/1005.4496, Apr. 2010.

[16] Leonid Portnoy,Eleazar Eskin and Sal Stolfo *Intrusion detection using Unlabeled Data using Clustering*,Columbia University.

[17] YUAN-CHENG LI, ZHONG-QIANG WANG, *AN INTRUSION DETECTION METHOD BASED ON SVM AND KPCA* ,Proceedings of the 2007 International Conference on Wavelet Analysis and Pattern Recognition, Beijing, China, 2-4 Nov. 2007.

[18] KDD CUP 99 data, *http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html*