International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-2, Issue-11 E-ISSN: 2347-2693

Self-Learning and Configurable IDS for Dynamic Environment

Manish Kumar[1*] and M. Hanumanthappa[2]

[1*] *Dept. of Master of Computer Applications,*
M. S. Ramaiah Institute of Technology, Bangalore, Bangalore University, Bangalore, INDIA
[2] *Dept. of Computer Science and Applications,Jnana Bharathi Campus, Bangalore University,Bangalore -INDIA*

www.ijcaonline.org

Received: Oct/22/2014 Revised: Nov/04/2014 Accepted: Nov/18/2014 Published: Nov/30/2014

Abstract— A major difficulty of any anomaly-based intrusion detection system is that patterns of normal behavior change over time and the system must be retrained. One of the principal problems of the intrusion detection systems based on the anomaly detection principles is their error rate, both in terms of false negatives (undetected attacks) and false positives, i.e. legitimate traffic labeled as malicious. This problem is amplified by the fact that the sensitivity (and consequently the error rate) varies dynamically as a function of the network traffic. An IDS must be able to adapt to these changes, and be able to distinguish these changes in normal behavior from intrusive behavior. In this paper, we address some of the key issues of detecting intrusion when a potential change occurs in operational environment and learn from the changed environment.

Keywords— Network Intrusion Detection System (NIDS), Stream Data Mining, Drift Detection, Early Drift Detection Method (EDDM)

I. INTRODUCTION

A major shortcoming of current IDSs that employ data mining methods is that they can give a series of false alarms in cases of a noticeable systems environment modification. There can be two types of false alarms in classifying system activities in case of any deviation from normal patterns: false positives and false negatives. False positive alarms are issued when normal behaviors are incorrectly identified as abnormal and false negative alarms are issued when abnormal behaviors are incorrectly identified as normal. Though it's important to keep both types of false alarm rates as low as possible, the false negative alarms should be the minimum to ensure the security of the system. To overcome this limitation, an IDS must be capable of adapting to the changing conditions typical of an intrusion detection environment. Specifically the Network Intrusion Detection System (NIDS) which detect the anomaly based on the network profile which is a streaming data and very much dynamic in nature.

One of the challenges of network streaming data classification is that the underlying data generation process of a stream tends to change over time, which is called as concept drift. A model learned from an earlier part of stream loses its classification accuracy upon the arrival of new instances that exhibit concept drift. As a result, any classification algorithm that is to be applied to data streams must be adjusted to effectively detect the concept drift when it occurs, and efficiently update the classification model to reflect the new concept.

Corresponding Author: *Manish Kumar*
[1*]*Dept. of Master of Computer Applications, M. S. Ramaiah Institute of Technology, Bangalore, Bangalore University, Bangalore, INDIA*

II. PROBLEMS RELATED TO IDS'S SELF LEARNING AND CONFIGURABILITY

System security administrators can tune the IDS by adjusting the profile, but it may require frequent human intervention. Since normal system activities may change because of modifications to work practices, it is important that an IDS should have automatic adaptability to new conditions. Otherwise, an IDS may start to lose its edge. Such adaptability can be achieved by employing incremental mining techniques. Such an adaptive system should use real time data to constantly update the profile. One straightforward approach can be to regenerate the profile with the new audit data. But this would not be a computationally feasible approach. When the current usage profile is compared with the initial profile, there can be different types of deviation. Each of these deviations can represent an intrusion or a change in behavior. In case of a change in system behaviors, the base profile must be updated with the corresponding change so that it does not give any false positives alarms in future. This means that the system needs a mechanism for deciding whether to make a change or reject it. If the system tries to make a change to the base profile every time it sees a deviation, there is a potential danger of incorporating intrusive activities into the profile. The IDS must be able to adapt to these changes while still recognizing abnormal activities. If both intrusive behavior and change in normal behavior occur during a particular time interval, the problem becomes more complicated. Again, determining which rules to add and which to remove is critical. There are also additional issues that need to be addressed in case of updating. The system should adapt to rapid changes as well as gradual changes in system behavior. Selecting the time interval at which the update should take place is also an important issue. If the interval is too long, the system may

miss some rapid changes or short-term attacks. If the interval is too small, the system may miss some long-term changes.

Since the traffic pattern is dynamic in nature and evolve with time, the IDS needs to be retrained in order to reduce the sudden rise in false positives or false negative. Hence, new training data needs to be acquired and it should contain sufficient samples of the new data pattern. Ideally the new training data samples should be attack-free to reduce any risk of a malicious user using a retraining opportunity to poison the training data set. During the retraining, the IDS also has to determine how to incorporate the new traffic pattern. Completely eliminating the previous pattern can result in having to constantly retrain the system in a case where cyclical traffic pattern changes occur. On the other hand, incorporating every observed traffic pattern can lead to a bloated database that suffers from a higher false positives or false negative rate since its pattern database is too general. Self-learning, adaptability and tuning must account for both micro and macro environment changes[15].

III. A FRAMEWORK FOR SELF-LEARNING AND CONFIGURABLE IDS

In this section we propose a framework for the adaptive maintenance of the profile rule set that can overcome the need for re-computation of the rules without sacrificing the detection capabilities.

The process begins with an initial set of audit data. Then stream rule mining will be applied to mine streaming network data. During each time window, the audit data in the incremental part will be mined and compared with the profile rule set. There can be three possibilities. If the similarity stays above threshold, no update is needed and the system continues with the current profile. If similarity goes below threshold with a sharp negative change, intrusion will be signaled and the profile will not be updated. If similarity goes below threshold with gradual change, the profile will be updated with the audit data in the current time window.

A. *Stream Data Mining*

The data stream paradigm has recently emerged in response to the continuous data problem. Algorithms written for data streams can naturally cope with data sizes many times greater than memory, and can extend to challenging real-time applications not previously tackled by machine learning or data mining. The core assumption of data stream processing is that training examples can be briefly inspected a single time only, that is, they arrive in a high speed stream, then must be discarded to make room for subsequent examples. The algorithm processing the stream has no control over the order of the examples seen, and must update its model incrementally as each example is inspected. An additional desirable property, the so-called anytime property, requires that the model is ready to be applied at any point between training examples.

I. *Mining Strategies*

Analogous to approaches used in data mining, there are two general strategies for taking machine learning concepts and applying them to data streams. The wrapper approach aims at maximum reuse of existing schemes, whereas adaptation looks for new methods tailored to the data stream setting.

Purposefully adapted algorithms designed specifically for data stream problems offer several advantages over wrapper schemes. They can exert greater control over processing times per example, and can conduct memory management at a finer-grained level. Common varieties of machine learning approaches to classification fall into several general classes. These classes of method are discussed below, along with their potential for adaptation to data streams: The following different modes of change have been identified in the literature:

- Concept change
  - Concept drift
  - Concept shift

Concept refers to the target variable, which the model is trying to predict. Concept change is the change of the underlying concept over time. Concept drift describes a gradual change of the concept and concept shift happens when a change between two concepts is more abrupt.

II. *Concept Change Detection in IDS*

Concept drift is defined as a change in the underlying data generation process. In the context of classification, concept drift is the change in statistical properties of the target variable, which the model is trying to predict, over time [2][6][7]. In this context, the term concept refers to the quantity we aim to predict. Three steps are required to handle a concept drift:

- **Monitoring Step**
- **Updating Step**
- **Diagnostic Step**

This kind of methods monitors the indicators of performance of the model such as accuracy, precision and recall (Klinkenberg, 2001). These indicators are monitored constantly and compared to a confidence level or an adjusted threshold. Two well-known performance-based methods can be cited: Drift Detection Method (DDM) proposed by Gama et al. (2004); and Early Drift Detection Method (EDDM) proposed by Baena-García et al. (2006). Next section describes both the Drift Detection Method (DDM) and Early Drift Detection Method (EDDM) in detail.

III. Drift Detection Method

Gama et al.[8][13] based their Drift Detection Method (DDM) on the fact, that in each iteration an live classifier predicts the decision class of an example. That prediction can be either true or false, thus for a set of examples the error is a random variable from Bernoulli trials. Let us denote p_i as the probability of a false prediction and s_i as its standard deviation calculated as given by Equation 1.

$$s_i = \sqrt{\frac{p_i(1-p_i)}{i}} \quad \text{..... Equation 1}$$

For a sufficiently large number of examples (n > 30), the Binomial distribution is closely approximated by a Normal distribution with the same mean and variance. For each example in the data stream the error rate is tracked updating two values: $p_{\min}$ and $s_{\min}$. These values are used to calculate a warning level condition presented in Equation 2 and an alarm level condition presented in Equation 3. Each time a warning level is reached, examples are remembered in a separate window. If afterwards the error rate falls below the warning threshold, the warning is treated as a false alarm and the separate window is dropped. However, if the alarm level is reached, the previously taught base learner is dropped and a new one is created, but only from the examples stored in the separate "warning" window.

$$p_i + s_i \geq p_{\min} + \alpha \cdot s_{\min} \quad \text{..... Equation 2}$$

$$p_i + s_i \geq p_{\min} + \beta \cdot s_{\min} \quad \text{..... Equation 3}$$

The values α and β in the above conditions decide about the confidence levels at which the warning and alarm signals are triggered. The value of $\alpha = 2$ and $\beta = 3$, gives approximately 95% confidence of warning and 99% confidence of drift.

DDM has good ability to detect abrupt and global drifts which affect the whole dataset. However, it presents low adaptability to gradual and local drifts which slowly affect some parts of the dataset. To overcome this problem we explored the Early Drift Detection Method (EDDM). The detail descriptions of EDDM is given in the next section.

IV. Early Drift Detection Method

The idea behind the Early Drift Detection Method (EDDM) is to consider the distance between two consecutive errors of classification. Notice that the error distance is represented by the numbers of instances between two consecutive classification errors. This approach assumes that if the distribution of the instances is stationary, the learning model will improve its prediction and the error distance will increase as the number of instances increases. Thus, a significant decrease in the error distance implies a drift. It will calculate the average distance between two errors ($p_i^{'}$) and its standard deviation ($s_i^{'}$). What it store are the values of $p_i^{'}$ and $s_i^{'}$ when $p_i^{'} + 2 \cdot s_i^{'}$ reaches its maximum value (obtaining $p_{\max}^{'}$ and $s_{\max}^{'}$). Thus, the value of $p_{\max}^{'} + 2 \cdot s_{\max}^{'}$ corresponds with the point where the distribution of distances between errors is maximum. This point is reached when the model that it is being induced best approximates the current concepts in the dataset.

The method defines two thresholds:

- $\left(p_i^{'} + 2 \cdot s_i^{'}\right) / \left(p_{\max}^{'} + 2 \cdot s_{\max}^{'}\right) < \alpha$ for the warning level. Beyond this level, the examples are stored in advance of a possible change of context.
- $\left(p_i^{'} + 2 \cdot s_i^{'}\right) / \left(p_{\max}^{'} + 2 \cdot s_{\max}^{'}\right) < \beta$ for the drift level. Beyond this level the concept drift is supposed to be true, the model induced by the learning method is reset and a new model is learnt using the examples stored since the warning level triggered. The values for $p_{\max}^{'}$ and $s_{\max}^{'}$ are reset too.

The values of α and β has to be determined after some experimentation which is approx. 0.95 and 0.9 respectively.

If the similarity between the actual value of $p_i^{'} + 2 \cdot s_i^{'}$ and the maximum value $p_{\max}^{'} + 2 \cdot s_{\max}^{'}$ increase over the warning threshold, the stored examples are removed and the method returns to normality.

The EDDM method achieves an early detection in presence of gradual changes, even when that change is very slow. This method shows a way to deal with noisy datasets even when the base algorithm is not designed with that aim.

Though the EDDM is better than DDM but most of these concept drift detection methods detect drifts by measuring the changes in the whole instance space. Such drift detection methods are called as global drift detection methods. A limitation of global drift detection is that if the data undergo a partial drift in a small region, a global method may not be sensitive enough to detect it. To detect such partial drift detection methods has been discussed in [6] which we are implementing along with Early Drift Detection Method (EDDM). The next section briefly describe about the Partial Drift Detection Method.

V. Partial Drift Detection Method

A limitation of global drift detection is that if the data undergo a partial drift in a small region, a global method

may not be sensitive enough to detect it. Also, a global method may not be able to identify the regions where the change takes place and thus unable to update the model accordingly[1][3][5].

To detect partial drifts, we need to look into local regions of the instance space. One way to do so is to partition the instance space into subspaces and apply an existing global method to each of the subspaces. Such a strategy requires prior knowledge of how the instance space should be partitioned. A coarse partition may not offer much of the benefits of partial detection, while a fine partition may be too sensitive to noise and data variation. To overcome these problems, a partial drift detection method based on a rule induction framework is described below.

1 Apply a rule induction method to the current available data (e.g., the first chunk of data) to learn a set of classification rules.
2 When a new data chunk is available, detect partial drifts as follows:

a) For each rule whose coverage is over a user-specified threshold, apply a drift detection measure to detect changes in the local instance space covered by the rule.
b) If no rule has coverage over the user-specified threshold, apply the same measure to each rule whose coverage is over the 90th percentile of the rule coverage in the model to detect drifts in the local region covered by the rule.

In this procedure, only rules with good coverage are used to identify regions for local drift detection. The reason for such a choice is that changes in a small region may be well due to noise or data variation and thus are not reliable indicators of concept drifts. In addition, rules with good coverage usually describe the concept in the data better than the rules that cover few examples. A significant change of class distribution in the region covered by such a rule is a better indicator that the concept is changing. The coverage threshold can be user-specified (such as 10%), or a high percentile of the rule coverage values. We use a user-specified threshold first. If none of the rules has coverage over the threshold (meaning all the rules have small coverage), we choose all the rules whose coverage is over the 90th percentile. The reason for this second threshold is to choose rules with relatively high coverage from low-coverage rules; otherwise, no rules can be used for drift detection. The choice of the 90th percentile is based on experimental results [6].

VI. Heuristics for Forgetting

When using a learner with partial instance memory to acquire concepts that change, there must be a mechanism to remove irrelevant examples of the old concept. Widmer and Kubat [9] investigated a heuristic approach to size such a window dynamically. Their window adjustment heuristic algorithm (Figure 1) takes into account current performance, whether accuracy is decreasing, and the coverage of the current concept descriptions. The user must set three parameters that determine thresholds for acceptable coverage and accuracy, but provided that concepts do not change too frequently, this heuristic will, in principle, size a forgetting window irrespective of the periodicity of change.

The heuristic takes four actions: reduce the window's size by 20%, reduce it by one time unit, make no change, and increase it by one. When concepts change, signaled by low coverage or by poor and decreasing accuracy, the heuristic quickly decreases the size of the window by 20%. As the learner acquires new concepts, the heuristic makes no change to the window's size or increases it gradually [14]. When concepts are stable, signaled by high coverage and acceptable accuracy, the heuristic gradually decreases the window's size. We use this window's size to control the AVSpace in Rule Induction Algorithm (Figure 2).

Input:	***lc*: threshold for low coverage, user-defined** ***hc*: threshold for high coverage, user-defined** ***p*: threshold for acceptable accuracy, user-defined** ***N*: examples covered by the positive concept description** ***S*: number of conditions in the positive description** ***Acc*: accuracy of current concept descriptions**
Output:	***w*: window size**

if $(N/S < lc \vee (Acc < p \wedge decrea\sin g(Acc)))$
  $\Delta w = -0.2w;$
else if $(N/S > 2.0 \times hc \wedge Acc > p)$
  $\Delta w = -1.0;$
else if $(N/S > hc \wedge Acc > p)$
  $\Delta w = 0.0;$
else
  $\Delta w = 1.0;$
$w = w + \Delta w;$
end

Figure 1:- *Heuristic for Dynamically Adjusting the Window Size*

IV. BUILDING AND UPDATING IDS RULESET USING EARLY DRIFT DETECTION AND PARTIAL DRIFT DETECTION

The approach is a general framework for rule learners, and any rule learning algorithm can be utilized for this approach. The learning, pruning and classification routine of a rule

learner does not need to be changed but the rule learner needs to be augmented to support rule quality, which is a trivial task.

We have used slightly modified version of SRLF Algorithm [6][15] for our work. The modified algorithm is based on Early Drift Detection (EDDM) which we have named as Rule Induction Algorithm (Figure 2). In this algorithm, we first learn a set of rules from the first chunk of data, and calculate rule qualities on that chunk using one of the rule quality measures [10][11][12]. When a new chunk of data is available, the Early Drift Detection method is used on rules with relatively high coverage. If a drift is detected on the region covered by a rule, we compute the qualities of this rule and the rules that were not used in drift detection (i.e., the rules with low coverage) on the new data. If the quality of a rule drops on the new data, which suggests the negative impact of the drift on the rule performance, the rule is removed from the model. Unaffected rules remain part of the classification model. Finally, if a drift is detected, a new model is learned from the new and past relevant data and added to the current classification model.

The strategy of keeping previously-learned consistent rules ensures that some of the past history is emphasized within the new model. If future concepts overlap with the current concept, which is the case in partial drifts, inclusion of these rules can improve the classification accuracy of the new model [7] [11][13].

A. *Evaluation*

We compared the performance of our drift detection approach in classification with other concept drift detection methods. In classification problems, we compare Partial Drift Detection (PDD) to two known methods for concept Drift Detection: DDM (Gama et al., 2004), Early Drift Detection (EDDM) (Baena-Garcıaet al., 2006).

B. *Experimental Datasets (KDD Cup 99 Dataset)*

We have used the 10% version of the KDD Cup 99 Dataset, which is more concentrated. This data set was used in KDD Cup 1999 Competition. The full dataset has about five million connection records, this is a set with only 10 % of the size. Here different classes appear and disappear frequently, making the new class detection challenging. This dataset contains TCP connection records extracted from LAN network traffic at MIT Lincoln Labs over a period of two weeks. Each record refers to either to a normal connection or an attack. There are 22 types of attacks, such as buffer-overflow, portsweep, guess-passwd, neptune, rootkit, smurf, spy, etc. So, there are 23 different classes of data. Most of the data points belong to the normal class. Each record consists of 42 attributes, such as connection duration, the number of bytes transmitted, number of root accesses, etc.

Input:	**Network data chunk**	**D**
	AVSpace (maintaining relevant instances)	Λ
	Rule-based classifier	**C**
	Early Drift Detection Method	**M**
	Rule coverage threshold	θ
Output:	**Updated Classifier**	**C**
	Updated AVSpace	Λ

```
1:  if  C == NULL then
2:      insert instance of D into Λ
3:      C ← classifier learned from Λ
4:  else
5:  For each rule R in C do
6:      if coverage(R) > θ then
7:        detect drift over the region covered by R using M
8:      end if
9:  end for
10: if no rule has coverage > θ then
11:     for each rule R in C do
12:         if coverage(R) > the 90th percentile then
13:             detect drift over the region covered by R using
M
14:         end if
15:     end for
16:  end if
17: if drift is detected then
18:    for each rule R in C do
19:        if drift is detected over R or coverage(R) is below
the
           threshold then
20:        Q_old ← rule quality from Λ
21:        Q_new ← rule quality from D
22:              if Q_old > Q_new then
23:               remove R from C
24:                     if drift is detected over R then
25:                        remove positive instances
                           covered by R from Λ
26:                     end if
27:              end if
28:       end if
29:    end for
30: end if
31: Insert instances of D into Λ
32: if drift was detected then
33:       C ← C ∪ classifier learned from Λ
34: end if
35: end if
36: return C and Λ
```

Figure 2:- *Rule Induction Algorithm*

C. *Performance Analysis*

Performance comparisons are done by evaluating detection quality and error rate. A True Positive (TP) detection is defined as a detection within a fixed delay range after the precise concept change time. A False Negative (FN) is defined as missing a detection within the delay range, and a False Positive (FP), as a detection outside this range or an extra detection in the range. The detection quality (DQ) is measured by TP/(TP+FN) and Precision(P)=TP/(TP+FP) of the detector.

Algorithm	FN	FP	TP	DQ	P
DDM	2.75	1.99	96.9	0.97240341	0.97987663
EDDM	1.75	1.43	97.1	0.98229641	0.98548665
PDD	0.82	0.98	98.1	0.99171047	0.990109

Table 1:- *Comparative Analysis of Drift Detection Methods*

The results are presented in the Table 1, which shows that the performance of partial drift detection is better than the DDM and EDDM. The False Positive and False Negative both the values are less for partial drift detection (Figure 3) and precision is comparatively high.

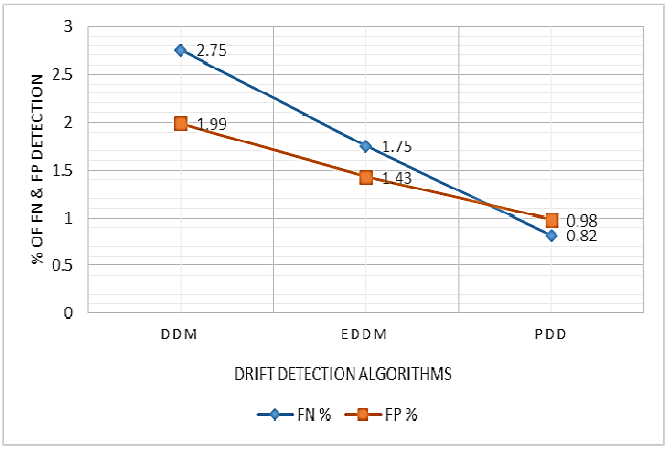

Figure 3:- *Comparative Analysis of False Negative and False Positive Detection*

V. CONCLUSION

In this paper, we have addressed the issues of detecting intrusion with high precision when a potential changes occurs in operational environment. Since the traffic pattern is dynamic in nature and evolve with time, the IDS needs to be retrained in order to reduce the sudden rise in false positives or false negative. Our experiments shows that the proposed scheme is more robust to the changing environment and has better precision. Experimental results show that the method has high precision, and performs well to reduce the False Negative and False Positive rate. The primary goal of devising a solution to decrease human effort for reconfiguring and tuning the IDS for a changed environment was accomplished.

REFERENCES

[1] A. Asuncion and D. J. Newman. UCI Machine Learning Repository [http://www.ics.uci.edu/_mlearn/mlrepository.html]. University of California, Irvine, School of Information and Computer Sciences, 2007.

[2] Albert Bifet and Richard Kirkby Data Stream Mining A Practical Approach :August 2009.

[3] Andrei Bara, Prof. Wayne Luk, "DeADA Self-adaptive anomaly detection dataflow architecture, Master's thesis, Master of Engineering in Computing of Imperial College London,2013.

[4] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. On demand classification of data streams. In Knowledge Discovery and Data Mining, pages 503–508, 2004.

[5] Concept drift - http://en.wikipedia.org/wiki/Concept_drift.

[6] Damon Sotoudeh, Aijun An, "Partial Drift Detection Using a Rule Induction Framework", CIKM'10 Proceedings of the 19th ACM International Conference on Information and Knowledge Management, Pages 769-778, 2010

[7] Dariusz Brzezinski, "Mining Data Streams with Concept Drift" , Poznan University of Technology, Faculty of Computing Science and Management, Institute of Computing Science,2010.

[8] Fredrik Gustafsson. Adaptive Filtering and Change Detection. Wiley, 2000.

[9] G.Widmerand M.Kubat. Learning in the presence of concept drift and hidden contexts. Machine learning, 23(1):69–101,1996.

[10] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Asurvey on sensor networks. IEEE Communications Magazine, 40(8):102–116, 2002.

[11] Leo Breiman. Rejoinder to discussion of the paper "arcing classifiers". The Annals of Statistics, 26(3):841–849, 1998.

[12] Maayan Harel, Koby Crammer, Ran El-Yaniv, Shie Mannor, "Concept Drift Detection Through Resampling", Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

[13] Manuel Baena-Garc´ıa, Jose´ del Campo-A´ vila, Rau´ l Fidalgo, Albert Bifet, Ricard Gavald´a, and Rafael Morales-Bueno. Early drift detection method. In Fourth International Workshop on Knowledge Discovery from Data Streams, 2006.

[14] Marcus A. Maloof, "Incremental Rule Learning with Partial Instance Memory for Changing Concepts", Proceedings of the 2003 International Joint Conference on Neural Networks, 2764–2769. Los Alamitos, CA: IEEE Press

[15] Thomas G. Dietterich. Machine learning research: Four current directions. The AI Magazine, 18(4):97–136, 1998.

AUTHORS PROFILE

Manish Kumar is working as Assistant Professor in the Department of Computer Applications, M. S. Ramaiah Institute of Technology, Bangalore, India. He is pursuing his PhD from Bangalore University, Bangalore. His specialization is in Network and Information Security. He has worked on the R&D projects related on theoretical and practical issues about a conceptual framework for E-Mail, Web site and Cell Phone tracking, which could assist in curbing misuse of Information Technology and Cyber Crime. He has published many research papers in National, International Conferences and Journals. He is also the active member of various professional societies.

Dr. M Hanumanthappa is currently working as Professor in the Department of Computer Science and Applications, Bangalore University, Bangalore, India. He has over 15 years of teaching (Post Graduate) as well as Industry experience. He is member of Board of Studies /Board of Examiners for various Universities in Karnataka, India. He is actively involved in the funded research project and guiding research scholars in the field of Data Mining and Network Security. He has published many research papers in National, International Conferences and Journals. He is also the active member of various professional societies.