# Time Critical Intrusion Detection System

*T.P.Saravanabava*[a] **and** *P.Narayanasamy*[b],
[a] Department of Electrical and Electronics Engineering
[b] Department of Computer Science &Engineering
Anna University
Chennai, India

Summary
. Communication system to support real-time system must provide transparent protection, better adaptation to changing environments, assured availability of critical services, timing and performance guarantees and the protection against disruption through naturally occurring events or malicious attack. Most of the thrust in today's IDS research is focused on accurate detection of attacks. Time critical real-time application demands the detection of attacks at a minimal, uniform and predictable time. This paper analysed, designed and implemented a parallel implementation of intrusion detection rules learned using RIPPER algorithm. We tested its performance using KDD CUP 1999 intrusion detection data. Obtained accuracy and time performance were analysed.

***Key words:***
*Intrusion Detection, Real-time applications, RIPPER*

## 1. Introduction

Recent advances in technology and human's thrust to reach the heather too unexplored frontier let to situations which forces us to be prepared for several new challenges. As an example, stationing scientist in the space for an extended period for the purpose of research and soldiers in the front line of battlefield poses very peculiar challenges. In these situations, if scientist or soldiers need urgent medical attention at an affordable cost at the quickest possible time, real-time systems come into help. It is demonstrated successfully that surgeons were able to perform an operation on a patient 7000 KM away with the help of robots and dedicated high-speed low latency communication infrastructure. This is to become cost effective so as to reach small far away communities and hostile places, technology should allow the usage of secure and reliable shared networks which will provide the same very low latency with high security. Main aim of our research is to speed up of operation without sacrificing accuracy of the total system. We have selected for this task a soft computing technique RIPPER which provides for excellent opportunity to parallelize the prediction operation and there by allowing the overall performance improvement in the resultant real-time system. The particular case, which we have selected for our study for the purpose of system requirement parameters and to rank the perceived threat to such real-time system, is tele-surgery.

Since these systems are particularly very sensitive to delay, delay variations and bandwidth variations. These parameters must be guaranteed. At the same time we must come up with a strategy, which will satisfy these requirements with a least cost. So, the ultimate choice leads to use of shared data network, which will guarantee the necessary bandwidth. Since, these real-time systems are very sensitive and cannot be suspended at the middle, all systems must provide the desired service through out the procedure. So, we propose a system in which at least 2 layers of communication options are kept ready one is a shared network that would be primarily used and another stand by dedicated network, which will be utilised in case the primary shared network fail to meet the system requirement at any point of time of the procedure.

We propose an end-to-end encrypted tunnel from the surgeon's computer to the computer, which directly connects to the robots, which perform the procedure under the guidance of the surgeon. This tunnel should be established by authenticating the both end systems and applications at the both ends. All communication must be encrypted between these two systems, which will ensure both privacy, and integrity. Since, small deviation of robots hand movement may cause severe damage to the patient. So, integrity of the message transferred between the two ends must be protected. Single largest threat is the DOS and DDOS attacks which will eventually limit the available bandwidth between the two ends. This must be continuously monitored and responded quickly. For this we propose an intrusion detection model based on the RIPPER rule learning technique. We improved the classification part of the learned rules by parallelising it. Since, learning is a one-time process or a infrequent process compared to the classification operation. So, we considered in our research to improve the classification process alone. We studied and analysed our ideas using

the 1999 version of MIT Lincoln Laboratory – DARPA intrusion evaluation data. Our approach is to simulate the process using the DARPA data only the portion of classification of intrusions. Data collection and feature construction are required when applied in real networks. Architecture of our system is depicted in figure 1. Our analysis showed that it gives 1.8 times better performance for a system with two processors.

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices. Intrusion prevention is the process of performing intrusion detection and attempting to stop detected possible incidents. Intrusion detection and prevention systems (IDPS) are primarily focused on identifying possible incidents, logging information about them, attempting to stop them, and reporting them to security administrators. In addition, organizations use IDPSs for other purposes, such as identifying problems with security policies, documenting existing threats, and deterring individuals from violating security policies. IDPSs have become a necessary addition to the security infrastructure of nearly every organization Karen et al [6]. There are many types of IDPS technologies, which are differentiated primarily by the types of events that they can recognize and the methodologies that they use to identify possible incidents. These publications discusses the following four types of IDPS technologies:

- Network-Based, which monitors network traffic for particular network segments or devices and analyzes the network and application protocol activity to identify suspicious activity.
- Wireless, which monitors wireless network traffic and analyzes it to identify suspicious activity involving the wireless networking protocols themselves.
- Network Behavior Analysis (NBA), which examines network traffic to identify threats that generate unusual traffic flows, such as DDoS attacks, scanning, and certain forms of malware.
- Host-Based, which monitors the characteristics of a single host and the events occurring within that host for suspicious activity.

IDPSs typically record information related to observed events, notify security administrators of important observed events, and produce reports. Many IDPSs can also respond to a detected threat by attempting to prevent it from succeeding. They use several response techniques, which involve the IDPS stopping the attack itself, changing the security environment (e.g., reconfiguring a firewall), or changing the attack's content.

## 2. Related Work

Srinivas et al [5] describes approaches to intrusion detection using neural networks and support vector machines. The key ideas of their research are to discover useful patterns or features that describe user behaviour on a system, and use the set of relevant features to build classifiers that can recognize anomalies and known intrusions. Their observation shows that both neural networks and SVMs deliver accurate results and shows compatible level of performance. Whether to use SVMs or neural networks in implementing an intrusion detector depends on the particular type of intrusion (anomaly or misuse) that is under watch, as well as other security policy requirements. SVMs have great potential to be used in place of neural networks due to its scalability and faster training and running time. But SVMs can only make binary classification, which is a sevee disadvantage where the intrusion detection system requires multiple-class identification. On the other hand, neural network have already proven to be useful in many IDSs, and are especially suited for multi-category classification.

Sung et al. [4] presented a novel intrusion detection system that models normal behaviours with hidden Markov models and attempted to detect intrusions by noting significant deviations from the models. Neural network and fuzzy logic are incorporated into the system to achieve robustness and flexibility. Self-organizing map determines the optimal measures of audit data and reduces them into appropriate size for efficient modelling by HMM. Based on several models with different measures, fuzzy logic makes the final decision of whether current behaviour is abnormal or not. Experimental results with some real audit data showed that the proposed fusion produces a viable intrusion detection system.

Tatyana et al. [3] applied dynamic authorization techniques to support fine grained access control and application level intrusion detection and response capabilities to overcome the shortcoming of current intrusion detection systems that work in isolation from access control for the application the system aim to protect. Their argument is that this approach helps in cooperation and interoperation between these components which helps in early detection and response. They presented a generic authorization framework that supports security policies that can detect attempted and actual security breches and which can actively respond by modifying security policies dynamically. The GAA-API combines policy enforcement with application-level intrusion detection and response, allowing countermeasures to be applied to ongoing attacks before they cause damage. Because the API processes access control request by applications, it is ideally placed to apply application-level knowledge about policies and

activities to identify suspicious activity and apply appropriate response.

Suseela et. Al [2] presented a novel multilevel hierarchical Kohonen Net (K-Map) for an intrusion detection System. Each level of the hierarchical map is modelled as a simple winner-take-all K-Map. One significant advantage of this multilevel hierarchical K-map is its computational efficiency. Unlike other statistical anomaly detection methods such as nearest neighbour approach, K-means clustering or probabilistic analysis that employ distance computation in the feature space to identify the outliers, our approach does not involve costly point-to-point computation in organizing the data into clusters. Another advantage is the reduced network size. They used the classification capability of the K-Map on selected dimensions of data set in detecting anomalies. Randomly selected subsets that contain both attacks and normal records from the KDD cup 1999 benchmark data are used to train the hierarchical net. They use a confidence measure to label the clusters. Then they used the test set from the same KDD Cup 1999 benchmark to test the hierarchical net. They showed that a hierarchical K-Map in which each layer operates on a small subset of the feature space is superior to single-layer K-Map operating on the whole feature space in detecting a variety of attacks in terms of detection rate as well as false positive rate.

Sang et al [1] proposes in this paper, a novel intrusion-detection technique based on evolutionary neural networks (ENNs). Advantage of using ENNs is that it takes less time to obtain superior neural networks than when using conventional approaches. This is because they discover the structures and weights of the neural networks simultaneously. Experimental results with the 1999 Defense Advanced Research Projects Agency (DARPA) Intrusion Detection Evaluation (IDEVAL) data confirm that ENNs are promising tools for intrusion detection.

El-Moussa et al [7] proposed a new approach that deploys active routers within a network to provide a distributed and adaptable defence system. Each active router integrates firewall functionality, intrusion detection, and a cryptographic algorithm. The firewall and the intrusion detection are used to detect and block attack traffic coming from or going to a network. The active routers to provide a secure communication between end users on their behalf use the cryptographic algorithm. In addition, active routers use a dedicated active protocol to control the traffic passing through them, and to detect and to block the attack close to its origin. They have through simulation, demonstrated that their proposed architecture has the required functionality to defeat well-known attack types. Using a distributed approach overcomes the limitation of conventional techniques that deploy a single firewall or management station to protect an entire network. Each active router provides its own protection of its attached subnets and collectively they are able to offer a strong defence for the whole network. Even if an active router is directly connected to two subnets, it can still protect one subnet from an attacker coming from the other subnet. Should an active router become compromised then the others continue to protect the network. The adoption of an active router approach also allows each one to adapt in real time and reconfigure to block certain traffic profiles while allowing others to pass through. Using a distributed array of active routers also means that when an attack is detected then it can be traced back and blocked at the active router that is closest to the point of origin of the attack. Finally, the adoption of data encryption between active routers adds further protection against an attacker originating from within the network.

## 3. Background and Motivation:

### 3.1 Tele-surgery

A surgeon at a site remote from the patient performs tele-surgery, also called remote surgery. Surgical tasks are directly performed by a robotic system controlled by the surgeon at the remote site.

#### 3.1.1 Preceding technologies

Tele-surgery became a possibility with the advent of laparoscopic surgery in the late 1980s. Laparoscopy (also called minimally invasive surgery) is a surgical procedure in which a laparoscope (a thin lighted tube) and other instruments are inserted into the abdomen through small incisions. The internal operating field may then be visualized on a video monitor connected to the scope. In certain cases, the technique may be used in place of more invasive surgical procedures that require more extensive incisions and longer recovery times.

Computer-assisted surgery premiered in the mid-1990s; it was the next step toward the goal of remote surgery. The ZEUS Surgical System, developed in 1995 by Computer Motion, Inc., was approved by the Federal Drug Administration (FDA) in 2002 for use in general and laparoscopic surgeries with the patient and surgeon in the same room. ZEUS comprises three table-mounted robotic arms—one holding the AESOP endoscope positioner, which provides a view of the internal operating field, the others holding surgical instruments. The robotic arms are controlled by the surgeon, who sits at a console several meters away. Visualization of the operating field is controlled by voice activation, while the robotic arms are controlled by movements of the surgeon's hands and wrists.

Computer-assisted surgery has a number of advantages over traditional laparoscopic surgery. The computer interface provides a method for filtering out the normal hand tremors of the surgeon. Two- and three-dimensional visualization of the operating field is possible. The surgeon can perform a maneuver on the console, review it to be sure of its safety and efficacy, then instruct the remote device to perform the task. The surgeon is also seated in an ergonomic position with arms supported by armrests for the duration of the operation.

Operation Lindbergh

While the concept of tele-surgery seems like a logical technological progression—if a surgeon can perform a procedure from several meters away, why not from several thousand meters —there is a major constraint that could lead to disastrous results during surgery, namely time delay. In the case of computer-assisted surgery, the computer console and remote surgical device are directly connected by several meters of cable; there is therefore virtually no delay in the transmission of data from the console to the surgical device back to the console. The surgeon therefore views his or her movements on the computer interface as they are happening. If the surgical system were removed to a more distant site, however, it would introduce a time delay. Visualization of the operating field could be milliseconds or even seconds behind the real-time manipulations of the surgeon. Studies showed that a delay of more than 150–200 milliseconds would be dangerous; satellite transmission, for example, would introduce a delay of more than 600 milliseconds.

In order to make tele-surgery a reality, expert surgeons would need to work with the telecommunication industry to develop secure, reliable, high-speed transmission of data over large distances with imperceptible delays. In January 2000, such a project, labeled "Operation Lindbergh," began under the direction of Dr. Jacques Marescaux, director of the European Institute of Tele-surgery; Moji Ghodoussi, project manager at Computer Motions, Inc.; and communication experts from France Télécom. Testing began on a prototype remote system (a modified version of the ZEUS Surgical System called ZEUS TS) in September 2000, with data being relayed between Paris and Strasbourg, France—a distance of approximately 625 mi (1000 km). Once an acceptable length of time delay was established, trials began in July 2001 between New York City and Strasbourg.

On September 7, 2001, Operation Lindbergh culminated in the first complete remote surgery on a human patient (a 68-year-old female), performed over a distance of 4300 mi (7000 km). The patient and surgical system were located in an operating room in Strasbourg, while the surgeon and remote console were situated in a high-rise building in downtown New York. A team of surgeons remained at the patient's side to step in if need arose. The procedure performed was a laparoscopic cholecystectomy (gall bladder removal), considered the standard of care in minimally invasive surgery. The established time delay during the surgery was 135 ms—remarkable considering that the data travelled a distance of more than 8600 mi (14,000 km) from the surgeon's console to the surgical system and back to the console. The patient left the hospital within 48 hours—a typical stay following laparoscopic cholecystectomy—and had an uneventful recovery.

Since then, remote surgery has been conducted many times in numerous locations. To date Dr. Anvari, a laparoscopic surgeon in Hamilton, Canada, has conducted numerous remote surgeries on patients in North Bay, a city 400 kilometres from Hamilton. He uses a VPN over a non-dedicated fiberoptic connection that shares bandwidth with regular telecommunications data.

## 4. RIPPER

RIPPER was developed by William Cohen [13] based on repeated application of Fumkranz and Widmer's [14] IREP algorithm. (IREP stands for Incremental Reduced Error Pruning and RIPPER stands for Repeated Incremental Pruning to Produce Error Reduction.) The RIPPER algorithm represents a significant performance improvement over previous rule induction algorithms. For a training set of size n, RIPPER's performance scales as $O(n \log^2 n)$.

In order for objects to be classified by most learning algorithms, they must first be transformed into a representation suitable for concept learning [12]. All representations must consist of a vector of features, each describing some aspect of the objects to be classified. In most machine learning systems, a feature may be either nominal (including binary) or continuous. Nominal features are those that take one of a finite number of pre-defined values, whereas continuous features are those that take on integer or real numeric values.

In RIPPER, a decision rule is defined as a sequence of Boolean clauses linked by logical AND operators that together imply membership in a particular class. The clauses are of the form $A = x$ or $A \neq x$ for nominal attributes and $A \leq y$ or $A \geq y$ for continuous attributes and y is some value for A that occurs in the training set. A classification hypothesis is a sequence of rules usually ending in a default rule with an empty set of clauses. During classification, the left hand sides of the rules are applied sequentially until one of them evaluates to true, and then the implied class label from the right hand side of the rule is offered as the class prediction.

To explain{12] the operation of RIPPER, we consider the restricted case in which the examples fall into one of two classes: positive or negative. A high level view of the algorithm is presented in Figure 2, which shows the original steps in the IREP algorithm and the additions made in RIPPER. To the basic algorithm, RIPPER adds several rule optimization steps as well as the option to improve the rule set by repeating the entire process for a number of iterations. These additions will be discussed, but first an overview of the IREP algorithm is presented.

LOOP n TIMES
  Start with the empty rule (TRUE => positive).
  LOOP UNTIL the stopping condition is reached.
Partition the training set into a growing set and a pruning set.
Grow a rule by greedily adding a clause to the left hand side guided by the grow heuristic.
Prune a rule by greedily deleting sequences of final clauses guided by the prune heuristic.
Remove examples covered by the rule from the training set
  END LOOP.
  Perform rule optimization on the entire rule set.
END LOOP

Figure 2: High-level description of the RIPPER algorithm [12].

The original IREP algorithm forms rules through a process of repeated growing and pruning. During the growing phase the rules are made more restrictive in order to fit the training data as closely as possible. During the pruning phase, the rules are made less restrictive in order to avoid over fitting, which can cause poor performance on unseen examples. IREP splits the training examples into a growing set and a pruning set. Rules to predict the positive class are grown one at a time by starting with an empty rule and then adding clauses to the left hand side in a greedy fashion under the guidance of a grow heuristic. Growing of a single rule stops when it covers no negative examples from the growing set. Each rule is pruned immediately after it is grown by deleting clauses that cover too many negative clauses in the pruning set under the guidance of a prune heuristic. After a new rule is grown and pruned, the covered examples are removed from both the growing and pruning set. Then the remaining data is repartitioned and another rule is grown. This rule growing process continues until all the examples in the training set are covered or until some stopping condition is reached.

The bold-faced lines are new in RIPPER, while the rest of the algorithm is an implementation of IREP. During the rule growing phase, the goal is to add clauses greedily to an initially empty rule in such a way that the set of examples covered by the rule contains a maximum of positive examples and a minimum of negative examples. The grow heuristic used in RIPPER is the information gain function proposed by Quinlan [15]. The algorithm starts with a set To consisting of all examples remaining in the growing set. At the $i^{th}$ iteration of the rule growing algorithm, the learner is working with a set $T_i$ consisting of $t_i^+$ positive examples and $t_i^-$ negative examples. A measure of the information required to describe the class membership of all the examples is:

$I(T_i) = -\log_2 [t_i^+ / (t_i^+ - t_i^-)]$

The goal is to reduce the total amount of information. When a new clause $A_i$ is added to a rule, a new set of examples $T_{i+1}$ is formed consisting of all examples from $T_i$ covered by the new rule with $A_i$ added. Adding a clause can only restrict the coverage of a rule, therefore $T_{i+1}$ must always be a subset of $T_i$, though the new set may still contain both positive and negative examples. The information required to describe the new state is:

$I(T_{i+1}) = -\log_2 [t_{i+1}^+ / (t_{i+1}^+ - t_{i+1}^-)]$

If the addition of $A_i$ reduces the number of negative examples covered by the rule in relation to the number of positive examples, then the information required to describe the set will tend to decrease. A drop in the amount of information from $T_i$ to $T_{i+1}$ represents a gain in the amount of information contained in the rule under construction. But the goal is also to cover as many positive examples as possible, so the gain heuristic is defined as the number of remaining positive examples multiplied by the gain in information. So if $t_i^{++}$ of the positive examples from $T_i$ are still present in $T_{i+1}$ then:

$Gain(A_i) = t_i^{++}. (l(T_i) - l(T_{i+1}))$

After a rule is grown it will ideally cover many positive examples and no negative examples in the growing set However, for noisy data it is not always desirable for the rules to fit the particular idiosyncrasies of the training data too closely. Rules that overfit the training data may perform poorly on unseen examples. The function of the pruning stage, therefore, is to relax the rule so that it is more general and less prone to overfitting. In order to do this, the rule is tested against the examples in the pruning set If the rule overfit the growing set, it will cover negative examples from the pruning set The prune heuristic measures this coverage. The ideal is for the rule to cover many positive examples and no negative examples. Clauses are deleted to maximize the function:

$v=( p-n)/(p+n)$

where p and n are the number of positive and negative examples in the pruning set that are covered by the rule. Note that this value is maximized when n = O. In the original IREP implementations, clauses are deleted one by one in reverse order until no deletion can be found that increases the value of v. RIPPER extends this process to also consider dropping sequences of final clauses.

Finally, a stopping condition is used to decide when to stop adding rules to the hypothesis. Furnkranz and Widmer propose a heuristic of stopping when the error rate of the next rule is greater than that of the empty rule. In RIPPER this heuristic is replaced with one based on the Minimum Description Length (MDL) principle from information theory. This criterion, described in [16], is an elegant formula that balances accuracy against complexity based on the number of bits required to communicate complete and correct class information for a set of examples. The description length is obtained by adding the number of bits required to describe the classification hypothesis to the number of bits required to enumerate the exceptions to this hypothesis. Attempting to minimize this measurement will bias the learner towards simple and accurate rules. RIPPER stops adding rules when the new description length is more than 64 bits larger than the best description length so far.

As a refinement to the process described so far, RIPPER also includes two rule optimization steps. In the first step, each rule is considered in turn and two new potential replacement rules are grown. The first rule is grown and pruned starting with the empty rule as before and the second is grown starting with the original rule instead of the empty rule. The main difference is that both rules are grown and pruned so as to optimize the error rate of the entire rule set on the entire pruning set. After the new rules are formed, the decision on which of the three candidate rules to include in the hypothesis is guided by the MDL heuristic. Finally, RIPPER simply repeats the entire algorithm a number of times to try to cover any remaining positive examples. The number of iterations of this process is subject to a parameter set by the user. In the current experiments the default value of 2 is always used.

The description of RIPPER given above is for a two-class problem. RIPPER handles multiple classes by ordering them from least to most prevalent and then treating each in order as a distinct two-class problem. So if the classes are ordered $C_1$ to $C_k$, RIPPER first learns rules to distinguish the least prevalent class $C_1$ from classes C2 to Ck..Then all examples covered by these rules are removed, and RIPPER learns rules to distinguish $C_2$ from $C_3$ to $C_k$. This continues until only the most prevalent class $C_k$ remains and this is used as the default class [13]. Note that this only applies to non-overlapping classes. When classes overlap, the only sensible choice is to train one binary classifier for each class, so the effect of RIPPER's class ordering means that rules will always be learned to cover the positive class first. The negative class will be left as the default.

KDD Cup 1999 Data Set

The KDD Cup 1999 Intrusion detection contest data is used in our project. The data set contains 24 attack types [11]. These attacks fall into four main categories:

Denial of service (DOS):

In this type of attack an attacker makes some computing or memory resources too busy or too full to handle legitimate requests, or denies legitimate users access to a machine. Examples are Apache2, Back, Land, Mailbomb, SYN Flood, Ping of death, Process table, Smurf, Teardrop.

Remote to user (R2L):

In this type of attack an attacker who does not have an account on a remote machine sends packets to that machine over a network and exploits some vulnerability to gain local access as a user of that machine. Examples are Dictionary, Ftp_write, Guest, Imap, Named, Phf, Sendmail, Xlock.

User to root (U2R):

In this type of attacks an attacker starts out with access to a normal user account on the system and is able to exploit system vulnerabilities to gain root access to the system. Examples are Eject, Loadmodule, Ps, Xterm, Perl, Fdformat.

Probing:

In this type of attacks an attacker scans a network of computers to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use this information to look for exploits. Examples are Ipsweep, Mscan, Saint, Satan, Imap.

The data set has 41 attributes for each connection record plus one class label. R2L and U2R attacks don't have any sequential patterns like DOS and Probe because the former attacks have the attacks embedded in the data packets whereas the later attacks have many connections in a short amount of time. Therefore, some features that look for suspicious behavior in the data packets like numbers of failed logins are constructed and these are called content features (www.ll.mit.edu).

The 41 attributes are:

duration: length (number of seconds) of the connection.

protocol_type: type of the protocol, e.g. tcp, udp, etc.

service: network service on the destination, e.g., http, telnet, etc.

src_bytes: number of data bytes from source to destination.

dst_bytes: number of data bytes from destination to source.

flag: normal or error status of the connection.

land: 1 if connection is from/to the same host/port; 0 otherwise.

wrong_fragment: number of ``wrong'' fragments.

urgent: number of urgent packets.

hot: number of ``hot'' indicators.
num_failed_logins: number of failed login attempts.
logged in: number of ``compromised'' conditions.
root_shell: 1 if root shell is obtained; 0 otherwise.
su_attempted: 1 if ``su root'' command attempted; 0 otherwise.
num_root: number of ``root'' accesses.
num_file_creations: number of file creation operations
num_shells: number of shell prompts.
num_access_files: number of operations on access control files.
num_outbound_cmds: number of outbound commands in an ftp session.
is_hot_login: 1 if the login belongs to the ``hot'' list; 0 otherwise.
is_guest_login: 1 if the login is a ``guest'' login; 0 otherwise.
count: number of connections to the same host as the current connection in the past two seconds.
serror_rate: % of connections that have ``SYN'' errors.
rerror_rate: % of connections that have ``REJ'' errors.
same_srv_rate: % of connections to the same service.
diff_srv_rate: % of connections to different services.
srv_count: number of connections to the same service as the current connection in the past two seconds.
srv_serror_rate: % of connections that have ``SYN'' errors.
srv_rerror_rate: % of connections that have ``REJ'' errors.
srv_diff_host_rate: % of connections to different hosts.

Experiment and Result
We first trained the classifier RIPPER with full data set of 494021 records. Time taken for training is 156565 seconds. Classification of subset of data gave very good result of 99.99 percent. But this is not a realistic way of doing experiment. Since, same data is used for both training and testing.
Second time we took 75 percent data of 370514 records from the total of 494021 records for training. It took 99325 seconds for training. Then using the learned rules we tested the classification accuracy with remaining data. Test was conducted by taking 100%, 75%, 50% and 25% of remaining data and studied the accuracy and time taken for classification for normal method and parallel method. When 100% of remaining data, that is 25 percent of actual data were tested, obtained accuracy is 99.91%, 98.63%, 42.85% and 99.29% for corresponding DOS, PROBE, U2R, R2L and NORMAL as tabulated in Table 4. Total number of records in each of DOS, PROBE, U2R, R2L and NORMAL are 97866, 1028, 14, 288 and 24321. Number of misclassified records in each of these categories in the same order is 86, 14, 8, 2 and 74. By perusing the accuracy, for U2R we are getting less accuracy. But this occurs because number of records available for training and as well as testing is very meagre. This justifies the deficiency in the obtained rules for this class. We repeated the experiment with the parallel model, we obtained the improvement of 1.75 in the time taken to classify all these test cases compared to the Sequential single processor run. Similarly test was repeated for 75%, 50% and 25% of remaining data, that is 18.75%, 12.5% and 6.25% of actual data and obtained results were tabulated respectively in Table 3, Table 2 and Table 1. These figures also confirm the same line of improvement in classification time taken for our parallel method. Bar chart were plotted for each attack type showing the original time taken and the time for parallel run for different number of test cases as shown in figures 3, 4,5 ,6 and 7. Figure 3 and 4 for DOS and NORMAL traffic shows better improvement compared to other cases since they have more number of test cases. Similar results can be expected for other type of traffic also if number of test cases improves.

### 4.1 Test Result for 75% Training Data

| Attack Type | PROBE | U2R | R2L | NORMAL | DOS |
|---|---|---|---|---|---|
| Misclassified cases | 1 | 2 | 0 | 14 | 2 |
| Total test cases | 257 | 3 | 70 | 6080 | 24466 |
| % of Accuracy | 99.61 | 33.33 | 100 | 99.77 | 99.99 |
| Time with two processors | 6 | 2 | 3 | 122 | 473 |
| Time with one processor | 9 | 2 | 3 | 210 | 845 |

Table 1: Results with 25% of Test Data

| Attack Type | PROBE | U2R | R2L | NORMAL | DOS |
|---|---|---|---|---|---|
| Misclassified cases | 8 | 4 | 0 | 19 | 2 |
| Total test cases | 514 | 7 | 141 | 12160 | 48933 |
| % of Accuracy | 98.44 | 42.86 | 100 | 99.84 | 100 |
| Time with two processors | 11 | 2 | 4 | 230 | 919 |

| | | | | | |
|---|---|---|---|---|---|
| Time with one processor | 16 | 2 | 4 | 416 | 1665 |

Table 2: Results with 50% of Test Data

| Attack Type | PROBE | U2R | R2L | NORMAL | DOS |
|---|---|---|---|---|---|
| Misclassified cases | 14 | 4 | 0 | 70 | 13 |
| Total test cases | 771 | 10 | 212 | 18240 | 73399 |
| % of Accuracy | 98.18 | 60 | 100 | 99.62 | 99.98 |
| Time with two processors | 15 | 2 | 5 | 341 | 1368 |
| Time with one processor | 22 | 2 | 5 | 617 | 2478 |

Table 3: Results with 75% of Test Data

| Attack Type | PROBE | U2R | R2L | NORMAL | DOS |
|---|---|---|---|---|---|
| Misclassified cases | 14 | 8 | 2 | 74 | 86 |
| Total test cases | 1028 | 14 | 283 | 24321 | 97866 |
| % of Accuracy | 98.64 | 42.9 | 99.3 | 99.7 | 99.9 |
| Time with two processors | 22 | 2 | 6 | 461 | 1853 |
| Time with one processor | 32 | 2 | 6 | 823 | 3395 |

Table 4: Results with 100% of Test Data

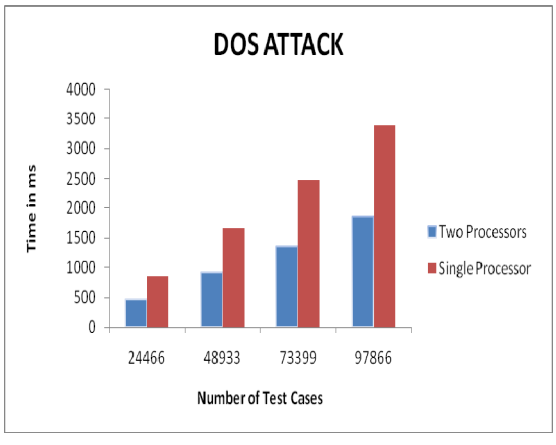


Figure 3: Time performance for detection of DOS Attack

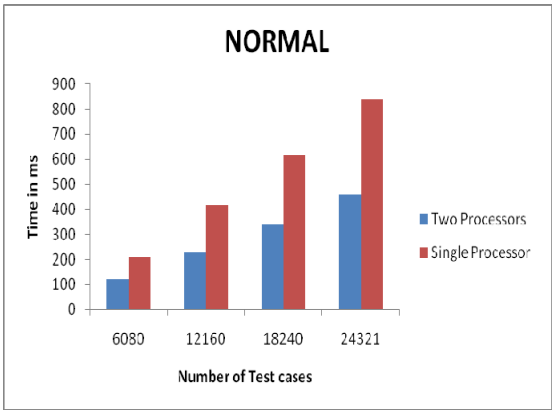


Figure 4: Time performance for detection of Normal traffic

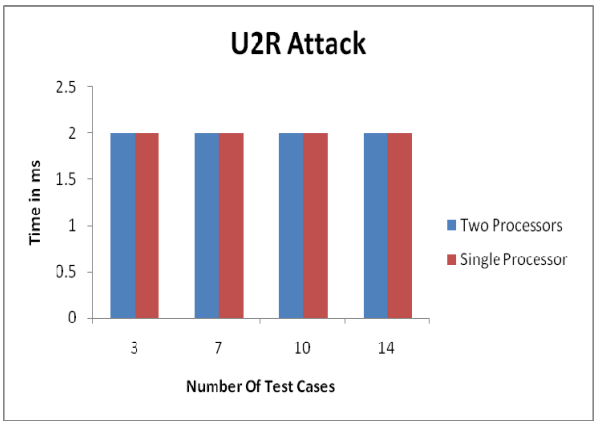


Figure 6: Time performance for detection of U2R Attack

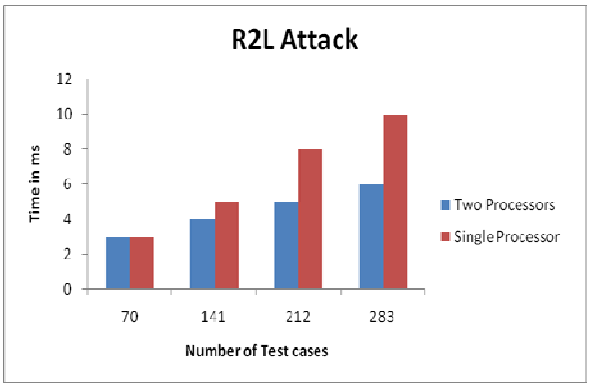


Figure 5: Time performance for detection of R2L Attack

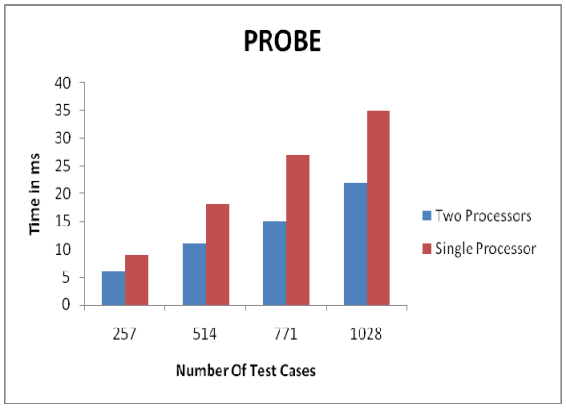


Figure 7: Time performance for detection of PROBE attack

## 4.2 Test Results for 50% Training Data

| Attack Type | PROBE | U2R | R2L | NORMAL | DOS |
|---|---|---|---|---|---|
| Misclassified cases | 1 | 0 | 30 | 34 | 0 |
| Total test cases | 513 | 6 | 141 | 12160 | 48932 |
| % of Accuracy | 99.8 | 100 | 78.7 | 99.7 | 100 |
| Time with two processors | 16 | 2 | 6 | 256 | 933 |
| Time with one processor | 23 | 2 | 6 | 463 | 1690 |

Table 5: Results with 25% of Test Data

| Attack Type | PROBE | U2R | R2L | NORMAL | DOS |
|---|---|---|---|---|---|
| Misclassified cases | 3 | 3 | 99 | 2127 | 8 |
| Total test cases | 1027 | 13 | 282 | 24320 | 97865 |
| % of Accuracy | 99.7 | 76.92 | 64.9 | 91.25 | 99.99 |
| Time with two processors | 21 | 2 | 9 | 529 | 1838 |
| Time with one processor | 30 | 2 | 9 | 949 | 3355 |

Table 6: Results with 50% of Test Data

| Attack Type | PROBE | U2R | R2L | NORMAL | DOS |
|---|---|---|---|---|---|
| Misclassified cases | 7 | 6 | 99 | 6721 | 8 |
| Total test cases | 1541 | 20 | 423 | 36480 | 146797 |
| % of Accuracy | 99.55 | 70 | 76.6 | 81.58 | 99.99 |
| Time with two processors | 29 | 2 | 9 | 675 | 2769 |
| Time with one processor | 42 | 2 | 9 | 1222 | 5069 |

Table 7: Results with 75% of Test Data

| Attack Type | PROBE | U2R | R2L | NORMAL | DOS |
|---|---|---|---|---|---|
| Misclassified cases | 9 | 10 | 126 | 8995 | 37 |
| Total test cases | 2055 | 27 | 564 | 48640 | 195730 |
| % of Accuracy | 99.56 | 62.96 | 77.6 | 81.51 | 99.98 |
| Time with two processors | 43 | 2 | 13 | 904 | 3644 |
| Time with one processor | 61 | 2 | 13 | 1637 | 6714 |

Table 8: Results with 100% of Test Data

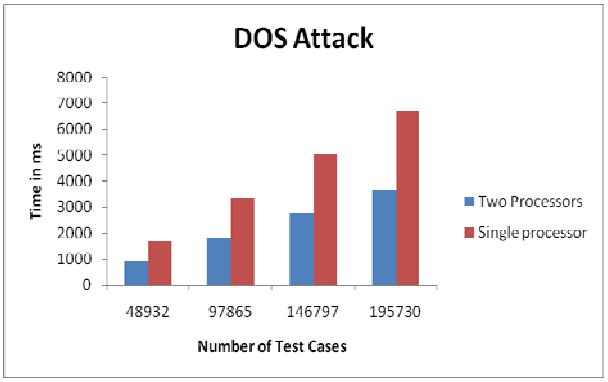


Figure 8: Time performance for detection of DOS attack

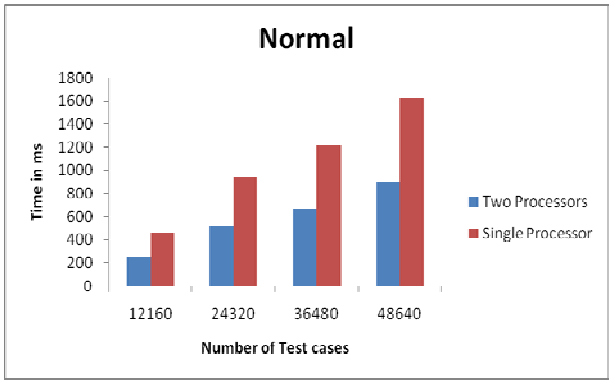


Figure 9: Time performance for detection of Normal Traffic

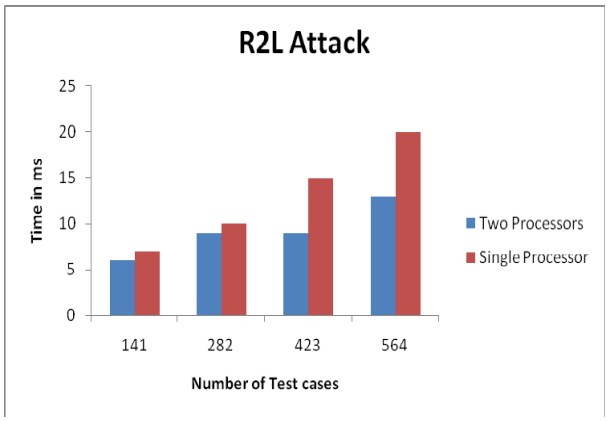


Figure 10: Time performance for detection of R2L attack

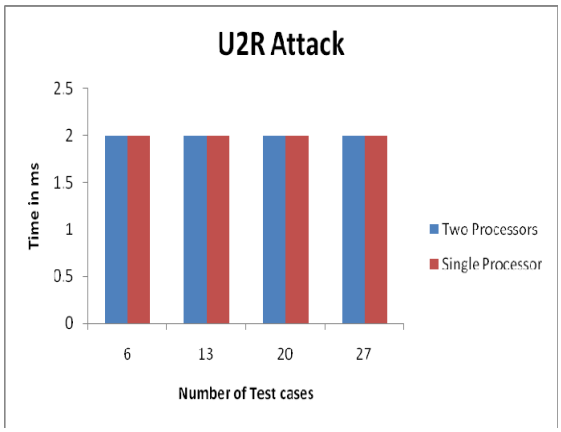


Figure 11: Time performance for detection of U2R attack

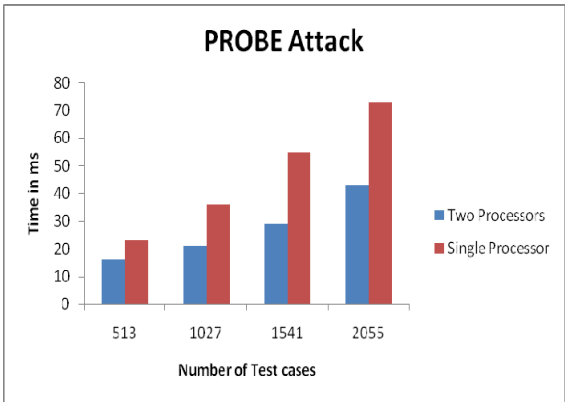


Figure 12: Time performance for detection of PROBE attack

Next we took 50 percent data of Total records numbering 247010 for training. It took 53895 seconds for training. Then using the learned rules we tested the classification accuracy with remaining data. Tests were conducted by taking 100%, 75%, 50% and 25% of remaining data and studied the accuracy and time taken for classification for normal method and parallel method as tabulated in tables 5, 6, 7 and 8. When 100% of remaining data, that is 50 percent of actual data were tested, obtained accuracy and time taken for the normal method and proposed method are tabulated above in table 8. Obtained results are comparable to the previous case. We repeated the experiment with the parallel model, we obtained the improvement of 1.8% in the time taken to classify all these test cases compared to the Sequential single processor run. This shows that when number of data to be processed increases, proposed method gives better performance. Graphically plotted the obtained gain in processing time for different type of attacks for different number of test cases for both the models for the easy visualization in figures 8 through 12.

## 5. Conclusion

As the result shows, our proposed method gives expected speed up to the intrusion detection process at the desired level. Results also shows attacks DOS and PROBE are detected more than 99 percent. Other attacks looks as if exhibiting poor performance but in reality considering the

number of cases available for training and testing indicates this is not a poor performance. As also considering the application, the computer used in these systems will not be providing any other service other than the specific service meant for the real-time system. So, perceived threat of U2R and R2L are small. More over, these specialised application will be available only with the institutions offering such a service and hence its exposure to large number of people for vulnerability analysis will be very limited. Considering all these factors our proposed system will serve the intended purpose at the desired time limit.

Other subsystem of the over all real-time system may also be studied for their enhancement in terms of improving its performance in achieving the desired result in a shorter and predictable time.

**REFERENCES**

[1] Sang-Jun Han and Sung-Bae Cho (2006) "Evolutionary Neural Networks for Anomaly Detection Based on the Behavior of a Program", IEEE transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, Vol 36, No. 3 Pg 559-570.

[2] Suseela T.Sarasamma, Qiuming A.Zhu, Julie Huff (2005), "Hierarchical kohonenen Net for Anomaly Detection in Network Security", IEEE transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, Vol 35, No. 2 Pg 302-160

[3] Tatyana Ryutov, Clifford Neuman, Dongho Kim, Li Zhou (2003) "Integrated Access Control and Intrusion Detection for Web Servers", IEEE Transaction on Parallel and Distributed Systems, Vol 14, No. 9 Pg 841-850.

[4] Sung-Bae Cho (2002) "Incorporating Soft Computing Techniques Into a Probabilistic Intrusion Detection System" IEEE transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews, Vol 32, No. 2 Pg 154-160

[5] Srinivas Mukkamala, Guadalupe Janoski, Andrew Sung (2002) "Intrusion Detection Using Neural Networks and Support Vector Machines", Proceedings of IEEE Intrnational Joint Conference on Neural Networks, pp. 1702-1701.

[6] Karen Scarfone, Peter Mell (2007) "Guide to Intrusion Detection and Prevention Systems (IDPS)" NIST Special publication 800-94.

[7] El-Moussa, F.A.; Linge, N.; Hope, M (2007)," Active router approach to defeating denial-of-service attacks in networks", IET Communications, Vol 1 pp. 55.63

[8] Jane W.S. Liu (2000), Real-Time Systems, Pearson Education

[9] St-Wr (2008) "Encyclopedia of Surgery: A Guide for Patients and Caregivers", http://www.surgeryencyclopedia. com/St-Wr/Telesurgery.html

[10] "Application - Revolutionary Telemedicine Techniques Lydia Dotto reports on Long-Distance Surgery", http://www.haivision.com/downloads/CSCmas.pdf

[11] MIT Lincoln Laboratory, "KDD cup 99 Intrusion detection data set.", http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz

[12] Sam Scott (1998) "Feature Engineering for a Symbolic Approach to Text Classification"', Master's Thesis, University of Ottawa, Ottawa, Ontario

[13] William W. Cohen. (1995) Fast Effective Rule Induction. In Proc. ICML-95. 1995. 115-123.

[14] Johannes Furnkranz and Gerhard Widmer. (1994) Incremental Reduced Error Pruning. Proc. ICML-94. 1994. 70-77.

[15] J. R. Quinlan. (1990) Learning Logical Definitions from Relations. Machine Learning 5:3. August, 1990. 236-266.

[16] J. Ross Quinlan and Ronald L. Rivest.(1989) Inferring Decision Trees Using the Minimum Description Length Principle. Information and Computation 80:3. March, 1989.227-248.

**About the Authors**

**P. Narayanasamy** received the Bachelor of Engineering Degree in Electrical and Electronics Engineering from Coimbatore Institute of Technology, Coimbatore, University of Madras, India in 1980, Master of Engineering Degree in Electrical and Electronics Engineering in 1982 and PhD Degree in the area of Computer Applications in Electrical Engineering in 1989 from Anna University, India. He is currently Professor and Head of the Department of Computer Science and Engineering, Anna University, Chennai, India. His research interests include VLSI Design and Testing, Computer Communication Networks, Wireless and Mobile Computing. He is guiding many research scholars in these areas for MS and PhD Programmes. He has published many technical and research papers in the National and International Conferences and Journals.

**T.P. Saravanabava** is presently working as a Selection Grade Lecturer in the Department of Electrical and Electronics Engineering, Anna University Chennai, Chennai – 600 025, INDIA. He obtained his M.E. Degree in Computer Science and Engineering and currently undergoing his PhD in the area of Network Security and Intrusion Detection System in the Faculty of Information and Communication Engineering, Anna University Chennai, Chennai – 600 025, INDIA.