NETWORK INTRUSION DETECTION USING ITERATIVE HEURISTICS

BY

SAAD AHMED KHAN

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER ENGINEERING

MAY 2012

Network Intrusion Detection Using Iterative Heuristics

by

SAAD AHMED KHAN

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

In Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

IN

COMPUTER ENGINEERING

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

Dhahran, Saudi Arabia

May 2012

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by

Saad Ahmed Khan

under the direction of his Thesis Advisor and approved by his Thesis Committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER ENGINEERING

Thesis Committee

Dr. Sadiq M. Sait (Chairman)

Dr. Zubair A. Baig (Member)

Dr. Mohammed Sqalli (Member)

Dr. Basem AL – Madani
Department Chairman

Dr. Salam A. Zummo
Dean of Graduate Studies

12/1/13
Date

Dedicated to

My Parents

Acknowledgements

All sincere praises and thanks are due to Allah (SWT), for His limitless blessings on us. May Allah bestow his peace and blessings be upon his Prophet Mohammad (P.B.U.H) and his family. Acknowledgements are due to King Fahd University of Petroleum & Minerals for providing the computing resources for this research.

I would like to express my profound gratitude and appreciation to my thesis committee chairman Dr. Sadiq M. Sait for his guidance and patience throughout this thesis. I would also like to express my appreciation for Dr. Zubair A. Baig and Dr. M. H. Sqalli for their constructive comments. Their continuous support, advise and encouragement can never be forgotten. Also, I would like to express my deepest thanks to every instructor who contributed in building my knowledge and experience. Thanks are also due to faculty and staff members of Computer Engineering Department for their cooperation.

I also thank my beloved parents, and all my family members for their moral support throughout my academic career. I would also like to thank the research assistant community in KFUPM, especially Saif and others for their support.

Finally, thanks to everybody who contributed to this achievement in a direct or an indirect way.

Contents

List of Tables

List of Figures

THESIS ABSTRACT

Name: Saad Ahmed Khan

Title: Network Intrusion Detection with Iterative Heuristics

Major Field: COMPUTER ENGINEERING

Date of Degree: May 2012

Computer network security is becoming important day by day and with the advent of cloud computing its importance has increased manifolds. Network intrusion detection has always remained a fruitful solution for network security and a lot of research has been done in strengthening network intrusion detection techniques. Several statistical techniques have been used to develop rules for the system to detect the attacks. Genetic algorithms have been used to provide a set of rules which are effective in differentiating between malicious and normal connections. In the proposed intrusion detection system, the reverse engineered attack chromosomes represent malicious network connections and the NSL-KDD connection set is search based on them to identify malicious connections. A simulation based analysis of the proposed system with the results consolidating our findings are provided in this thesis work.Directions for future work specially for detection of zero day attacks is also provided.

Keywords: *Network Intrusion Detection, Computer security, Network security, Genetic Algorithm, Intelligent methods.*

MASTER OF SCIENCE DEGREE

King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia.

May 2012

نبذه مختصرة

الاسم: سعد أحمد خان

الموضوع: " كشف الاختراق للشبكات من خلال الاساليب البحثية "

المجال الرئيسي: " هندسة الحاسب "

تاريخ الدرجة: مايو 2012

أصبحت أمن شبكات الحاسب هامة يوماً بعد يوم مع ظهور الحوسبة السحابية زادت أهميته المتشعبة. ويبقي دائماً كشف الاختراق للشبكات حل مثمر في مجال أمن الشبكات وكثير من الابحاث نفذت بقوة في تقنيات كشف الاختراق للشبكات.

العديد من التقنيات الاحصائية استخدمت لتطوير قواعد لنظام كشف الهجمات. والخوارزميات الجينية المستخدمة لتقدم مجموعة من القواعد التي كانت مؤثرة بشكل كبير في التفريق بين الاتصالات الآمنة والاتصالات الخبيثة.

في هذا النظام المقترح لكشف الاختراق, فإن كروموسومات الهجوم المهندس المعاكس عبارة عن إتصالات خبيثة للشبكة ومجموعة اتصال "NSL-KDD" عبارة عن بحث قائم عليهم للتعرف علي هذه الاتصالات الخبيثة. نظام المحاكاه قائم علي تحليل النظام المقترح وبالنتائج المؤيدة لنتائجنا المقدمة في نبذة عملنا هذا.

فإن الاتجاهات في العمل المستقبلي خاصة في الكشف عن ايام خالية من الهجمات أيضاً قائمة.

كلمات البحث: كشف الاختراق للشبكات, أمن الحاسب, أمن الشبكات, الخوارزم الجيني, الطرق الذكية.

درجة الماجستير في العلوم

جامعة الملك فهد للبترول والمعادن, الظهران, المملكة العربية السعودية

مايو 2012

Chapter 1

Chapter Introduction

1.1 Intrusion

In computer and communication systems, an intrusion is referred to an attempt to break-in to an information system or performing an illegal activity. Intrusions are also referred to as anomalies, outliers, discordant observations, exceptions, aberrations, surprises, peculiarities, or contaminants in different application domains.

There are two general categories of intruders: external and internal. The external intruders are those who have no authorized access to the system and who attack by using various penetration techniques. On the other hand, the internal intruders have access (permission) to the system and they perform unauthorized activities.

1.2 Intrusion Detection Systems

Intrusion detection refers to the problem of finding patterns in a data-set (off-line environment) which do not conform to expected or normal behavior of the network or are known to be malicious in nature. Intrusion detection is of extensive use in a variety of applications such as fraud detection for credit cards, insurance and health care, malware detection and fault detection in computer networks and even in military surveillance for enemy activities. Intrusion detection techniques are different for different domains and applications.

An Intrusion Detection System (IDS) is installed as software or a physical device that monitors network traffic in order to detect unwanted activity and events and reports them accurately to the proper authority. Misuse detection and anomaly detection are two main classes of intrusion detection techniques.

Misuse detection models look for a pattern of malicious behavior, and behavior that fits this model is classified as malicious. Anomaly detection refers to the problem of finding patterns in data-set that does not conforms to expected or normal behavior. Anomaly detection is of extensive use in a variety of applications such as fraud detection for credit cards, insurance and health care, intrusion detection and fault detection in computer networks and even in military surveillance for enemy activities. Anomaly detection techniques are different for different domains and applications.

Intrusion detection techniques can be further classified into two categories based on where they look for malicious behavior. Network Based Intrusion Detection System (NIDS) and Host Based Intrusion Detection System (HIDS). Network based systems identify malicious behavior through network devices such as network interface cards and analyze network events such as protocol information, traffic volume, IP addresses, etc. where as host based systems monitor files, process identifiers, operating system related calls and activities related to specific hosts. Honeypots are among the widely used host based systems which are specifically designed to study intruders and intrusive activities. It appears as a target to the intruder and helps in tracking its location and responding to attacks [1].

The efficiency of a intrusion detection system is defined by the following three parameters:

1. **Speed:** The speed or the execution time refers to the time an IDS takes to decide whether the connection is anomalous or normal. Execution time is a very important parameter for any IDS. The faster a system detects an intrusion the more efficient it is and lesser is the damage caused by an attack. For non-real time systems where network log files are examined (e.g. for the past day), a system execution time of hours is acceptable; but in real time services this execution time is not acceptable. For real time applications, a quick system is required which detects an anomalous connection before it causes any kind of damage to the network system.

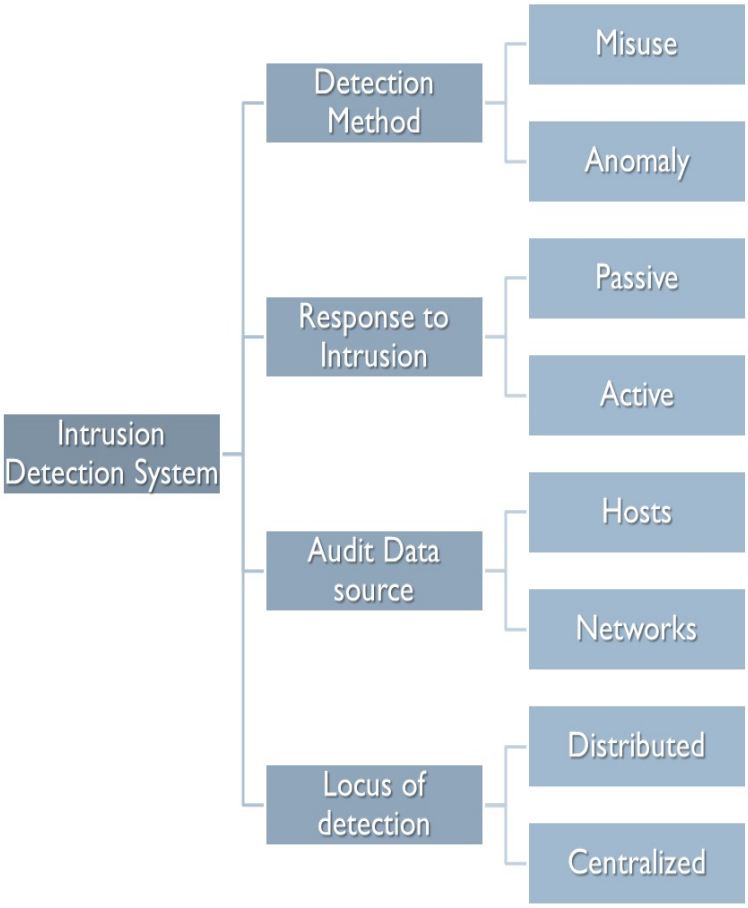

Figure 1.1: IDS characteristics

2. **Accuracy:** The accuracy of an IDS is a measure of how well a system classifies network connections. This classification rate depends on the number of true/false positives/negatives. The lesser the number of false alarms generated by the IDS the more accurate it is.

3. **Adaptability:** The adaptability of an IDS is its ability to adapt to detect new intrusions. It is a well known fact that new computer intrusions (attacks) evolve almost daily. Systems can guarantee perfect protection from the attacks

whose signatures are well known today. But for the new attacks of tomorrow if the system is not adaptable, the system will be vulnerable to those attacks and will not be able to ensure complete security. Therefore, the IDS should be able to adapt to protect against new attacks as well.

Figure 1.1 shows a more detailed classification of intrusion detection systems.

1.3 Genetic Algorithm

Genetic algorithm (GA) is a powerful domain independent search technique that is based on principles of evolution and natural selection. GAs were first introduced by John Holland and his colleagues in the early 1970s. In computer security applications, it is mainly used for finding optimal solutions to a specific problem. [2]
The process of a genetic algorithm usually begins with a randomly selected population of chromosomes. These chromosomes are representations of the problem to be solved. According to the attributes of the problem, different positions of each chromosome are encoded as bits, characters, or numbers. In addition, each chromosome is characterized by a fitness function which implies that chromosomes with higher fitness values will lead to a better solution making the selection process biased towards the fittest chromosomes. The fitness of a chromosome is calculated by an evaluation function or a fitness function. During evaluation, two basic operators, crossover and mutation, are used to simulate the natural reproduction and mutation

of species.[2, 3]

In the crossover process, two parents are combined to produce two offsprings. However, it is possible that the chromosomes of two parents are copied unmodified as offspring. There is also a possibility that the chromosomes of the two parents are randomly combined (crossover) to form an offspring. The mutation process randomly changes the value of a gene from its present state to an entirely distinct one. [2, 3]

1.4 Network Intrusion Detection and Genetic Algorithm

A network IDS collects traffic based information that traverses a network. This information is then fed to a rule generator which creates a set of rules for detection of attacks. As the requirement of our system is to develop a more evolved rule set for intrusion detection, a signature of a known attack is converted into a chromosome. A set of such chromosomes are provided to the system as initial population and the test set is searched based on it. In the first generation only attacks represented by the chromosomes of initial generation are detected. In the subsequent generation these chromosomes are evolved using crossover and mutation. With the evolved rule set, the intrusion detection system continues to function as before but it now has added capability to block the malicious network activity.

1.5 Thesis Objective

The objective of this thesis work is to develop a network intrusion detection scheme using reverse engineering for feature selection and an iterative heuristic such as genetic algorithm. The anomalous network connection are reverse engineered using feature selection methods and comparison with normal connections. The network intrusion detection system should be able to detect a good number of attacks and also should be able to keep the number of false alarms as low as possible.

1.6 Thesis Organization

The first chapter presents a general introduction. The second chapter covers the problem description. The third chapter presents a literature survey of intrusion detection systems. In the fourth chapter attacks of the NSL-KDD data set are explained. The fifth chapter explains the system implementation. In chapter six, feature selection techniques such as chisquare and filtereed attribute and filtered subset evaluation techniques are used for network intrusion detection. Chapter seven provides a detailed description on reverse engineering for feature selection and application of genetic algorithm in network intrusion detection. Finally in chapter eight the thesis in concluded and future directions are provided.

Chapter 2

Problem Description

2.1 Classification Problem

Classification is a problem of putting (classifying) each data instance or data record in a particular class. In a dataset the class is indicated by the goal attribute. The attributes can be discrete/continuous or symbolic values depending on the type of data and application domain. The data instances consist of two parts, a set of predictor attribute values and a goal attribute value. The goal attribute is usually the class of the data row. These attribute values are used to predict the class for unseen data. These data instances can be called an object, case, record, tuple or a connection. The attributes which define these records are called features or variables [4, 5]. For example if we have squares, triangles, pentagons and octagons in a sample space then the classification problem will be to identify the number of sides of the

shape and then put all the similar shapes under the same category. Then all the squares will be placed together. Similarly all the other shapes will be classified.

The classification problem involves two mutually exclusive data sets or instances, one is called the training data set which is completely made available to the data mining algorithms so that the algorithms have access to both the predictor attributes and the goal attribute values for each data instance. The other set of data is called the test set which may or may not contain the goal attribute value.

In intrusion detection all the attacks should be detected correctly so that the required action can be taken to limit or prevent their effect. On the other hand all the normal connections should be allowed to pass through the system un affected.

The aim of data mining algorithm is to develop or discover relationships between the predictor attributes and the goal attribute values using the training set only. The discovered relationships are then used to predict the class of all the data instances in the test set. On a labelled test set, the goal attribute is used to verify the accuracy of prediction for the classified data instances. If the predicted class is the same as the actual class, then the prediction is correct. And the prediction is not correct if the predicted class is different than the actual class of the data instance.

The main aim of the data mining algorithm is to maximize the classification accuracy in the test set, as achieving high classification accuracy on the training set can be considered as a trivial task. The prediction accuracy is defined as the number of correct predictions divided by the total number (correct + incorrect) of predictions.

$$PredictionAccuracy = \frac{Number\,of\,Correct\,Predictions}{Total\,Number\,of\,Predictions} \tag{2.1}$$

The discovered relationships are generally formulated as **IF-Then** rules. The conditional attributes become antecedent and the goal attributes constitute the consequents.

In a geometric representation, the goal of the classification algorithm is to find lines or curves that separate the data instances of the other classes. A simple example in a two dimensional space is shown below. As the number of independent variables in a data instance increase the number of dimensions in the geometric graph increase and the problem of finding those lines or curves become difficult. An example of this geometric representation is shown in figure 2.1

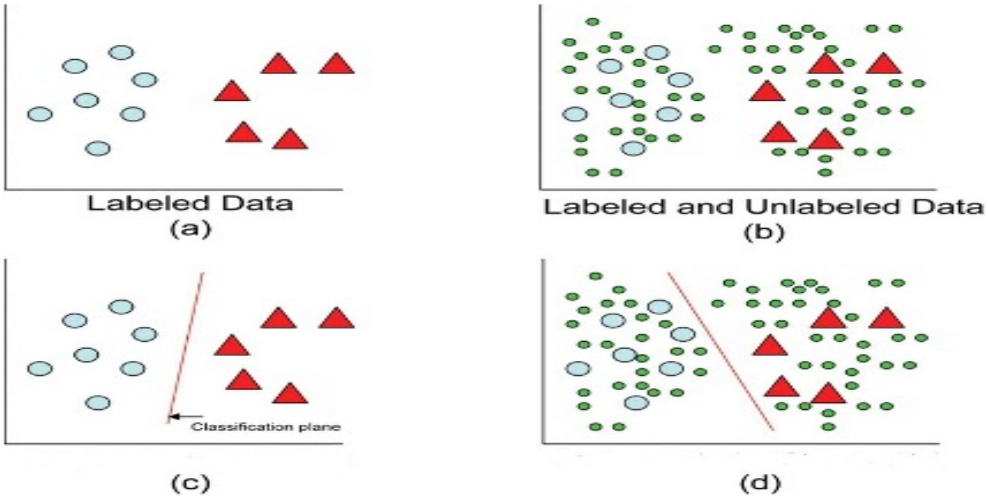

Figure 2.1: Classification geometric representation

The classification problem mainly comprises of following steps:

1. The data mining algorithm extracts some knowledge from the "training set".

2. The extracted rules are tested on the "test set"and the prediction accuracy is calculated.

3. Define how to classify data instances unknown to the algorithm i.e. data instances for which no relationships and rules were developed. A possibility is to use the existing data (training + test sets) as new training sets and develop rules to classify new data which was previously unknown.

2.2 Genetic Algorithm for rule discovery

In the genetic algorithm, each individual corresponds to a candidate solution to a given problem. In the problem of discovering prediction rules for intrusion detection, generally we are interested in a set of rules rather than a single golden rule. This is due to huge diversities in different types of attacks and their corresponding features. There are two different approaches for rule development with genetic algorithm.

1. **Pittsburg approach:** In this approach, each individual of the GA population represents a set of prediction rules, i.e. an entire candidate solution. The fitness function measures the rule set as a whole. This approach executes in a longer time because in any given generation there are more rules to be matched against the data set being mined.

2. **Michigan approach:** In this approach, an individual represents a single rule i.e. a part of the candidate solution. This approach is also referred to as classifier systems. In this approach, the fitness function measures the performance of a single rule out of the context of the other rules. This approach accounts for the discovery of a set of good rules rather than a single best rule.

2.2.0.1 Rule Encoding

When using genetic algorithms, the main task is the encoding of solutions. This is the part of the problem formulation and also is an important factor in the efficiency and results of the problem being solved. Following are the viable techniques for encoding:

1. **Binary Encoding:** All the parameters are encoded in s binary form. The technique depends on the number and kind of attributes and their corresponding values.

2. **Low level Encoding:** Attribute values are encoded directly into the genome.

3. **Threshold based encoding:** Attribute thresholds (upper, lower, or both) are encoded into the genome.

4. **Hybrid Encoding:** Encode the genome using the combination of the above mentioned techniques.

2.3 Problem Description

Intrusion detection falls into the category of classification problems where each network connection in the data set (in academic research), network log files (for passive systems) or real time environment has to be correctly classified as a malicious network connection (intrusion/attack) or a normal connection. Most of the approaches for intrusion detection aim to develop rules for intrusive connections and then search for a pattern match. The problem of intrusion detection is to develop such rules which accurately classify all the intrusions and normal connections.

As this thesis is targeted on the data set (KDD99/NSL-KDD) using genetic algorithm for rule (chromosome) evolution and reverse engineering for feature selection, the goal is to develop rules which correctly classify all the connections as per their classes. The detection rate in an IDS is synonymous to the classification rate in a data mining algorithm. For an IDS the following definitions are often used to describe the system efficiency:

1. **True positive (TP):**

   Correctly classifying an intrusion as an intrusion. The true positive rate is synonymous with detection rate, sensitivity, and recall, which are other terms often used in the literature.

2. **False positive (FP):**

   Incorrectly classifying a normal data connection as an intrusion. Also known

as a false alarm.

3. **True negative (TN):**

   Correctly classifying normal data connection as normal. The true negative rate is also referred to as specificity.

4. **False negative (FN):**

   Incorrectly classifying an intrusive or malicious connection as normal.

For a reliable intrusion detection system which correctly classifies all the connection the number of true positives and true negative should be high and the number of false positives and false negatives should zero (ideal case) or very small.
Using the above parameters the accuracy and precision of IDS is defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{No.\ of\ \ correct\ \ classifications}{Total\ \ no.\ \ of\ \ classifications} \tag{2.2}$$

$$Precision = \frac{TP}{TP + FP} = \frac{No.\ of\ \ \ correct\ \ intrusive\ \ classifications}{Total\ \ number\ \ of\ \ correctclassifications} \tag{2.3}$$

The goal of this thesis work is to build an intrusion detection system with high accuracy and precision along with the betterments in speed accuracy and adaptability as compared to other systems.

2.4 Research Approach

In order to achieve the objective of this thesis work, the following steps will be taken:

- Review the existing literature on the use of intelligent techniques based on iterative heuristics for network intrusion detection.
- Propose a genetic algorithm and a data mining based technique for model (rule) construction to accurately emulate intrusion data.
- Implement the technique using software based tools and perform simulations.
- Perform simulations with distorted and incomplete network intrusion data.
- Idealize the outcomes of the simulations through the selection of the most appropriate statistical techniques for attack parameter selection.
- Compare the results of the proposed technique in terms of the defined performance metrics.

Chapter 3

Literature Review

The anomaly detection techniques can be classified into the follwing main categories

1. Statistical-based techniques.
2. Knowledge based techniques.
3. Machine learning based techniques.
4. Evolutionary Algorithms

3.1 Statistical Based techniques

Statistical based techniques model profiles which represent the stochastic behavior of the network traffic activity. Traffic rate, number of packets for each protocol, connection rate, number of different IP addresses are among the parameters used

for profile modelling. These techniques use two sets of data. One set is the currently observed profile over time whereas the other is the previously trained profile. The current profile is compared with the previous one to determine a anomaly score which indicates the degree of irregularity for a specific event and is used to determine the occurrence of anomaly when this score exceeds a certain threshold [6].

Denning et al. [7] proposed univariate model and used the parameters as independent Gaussian random variables and defined an acceptable range of values for each parameter. Ye et al. [7] suggested use of multivariate models based on correlations between two or more metrics. Experiments showed better results with correlated data instead of individually selected values.

Statistical based techniques have several advantages over other techniques. Firstly they do not need to be trained with normal or malicious behavior of the network because they have the ability to learn normal traffic behavior from observations. Secondly, such systems prove to be useful in providing accurate information about malicious traffic over longer periods. Apart from the advantages, these systems are also exposed to the risk of being trained by the intruder to bypass the malicious traffic. Also defining thresholds is a difficult task and not all the network behaviors can be modelled with stochastic methods. The assumption made in these systems should be realistic.

3.2 Knowledge Based Techniques

Knowledge based schemes classify the presented data according a set of rules. Their operation involve three major steps. First a number of different attributes are identified and selected from the training data, second numerous classification rules/parameters are deduced and finally the presented data is classified according to the set of rules generated. Expert systems are an example of knowledge based systems. The model(rules) used is either human designed or is generated using some tool i.e. FSM or description languages. Est*é*vez-Tapiador et al. [8] presented one such technique. The advantage of knowledge based systems in intrusion detection is their robustness and flexibility. But the major drawback of such systems is their inability to generate perfect rules and models. This is because the task is very difficult and time consuming even for a human expert.

3.3 Machine Learning based Techniques

Machine learning based techniques aim to develop an explicit or implicit model that enables the patterns to be classified accordingly. Machine learning techniques require labelled data in order for the system to be trained to detect and classify connections accordingly. Some of the machine learning techniques are similar to statistical techniques in the way they improve their developed models by training with new data. The major drawback of such techniques is their heavy resource requirements. Many

machine learning techniques have been used to develop intrusion detection systems. Some of the techniques include Bayesian networks, artificial neural networks, fuzzy systems, Markov models and clustering for outlier detection.

3.4 Evolutionary algorithms based techniques

Several iterative algorithms such as genetic algorithms, particle swarm optimization, simulated annealing, taboo search, etc have been applied to the problem of intrusion detection. But genetic algorithm has been widely used mainly for intrusion detection in different flavors. In some cases a pure genetic algorithm has been applied whereas in other cases it is used in combination with other heuristics or techniques.

Genetic Algorithm or Genetic Programming based systems are superior as compared to others due to their ability to be easily retrained. It is sufficient to employ the best population evolved in the previous iteration as initial population and repeat the process, but this time including new data. This makes the system inherently adaptive [9].

However, the outcome produced by GA is known to be accompanied by a large amount of false positives. As such, most of the research in the area is centered on the concept of trying to minimize these false positives.

Chittur [10] proposed a genetic algorithm based technique which learns how to detect and separate malicious network connections from normal ones. This method

was supposed to eliminate the inherent problems of fingerprinting systems [10, 11]. The mathematical model decision tree held a randomized coefficient for the data, so that when it is multiplied by the data it yielded a certainty factor of weight which determined whether that record was an attack or not, the coefficients used were based on ERC (Ephemeral Redundancy Constants) [12]. The mutation operation was a result of slight change in these values. As the dataset is comprised of both symbolic and continuous values so different weights using ERC were selected for continuous values and for symbolic values the constants were randomly established. The fitness function used was based on how many attacks were correctly detected and how many legal connections were classified as attacks. The fitness function used is given in equation 3.1

$$Fitness = \frac{\alpha}{A} - \frac{\beta}{B} \tag{3.1}$$

where α is the number of attacks correctly detected, A is the total number of attacks, β is the number of normal connections classified as attacks (false negative) and B is the total number of normal connections. The generated mathematical model was then evolved using the genetic operators crossover and mutation based on the fitness.. The proposed technique was 97% accurate with a false positive rate of 0.6877%.

Li et al.[13] presented an initial framework for intrusion detection using genetic algorithms. The proposed technique models simple rules for network traffic that

differentiate normal connections from anomalous connections. Rules were stored in the form of "if **condition** then **act**". The condition refers to a match between the current network connection and the rules in the IDS, such as source and destination IP addresses and port numbers (used in TCP/IP network protocols), duration of the connection, protocol used, etc., and the act field defines the action based on the security policy such as such as reporting an alert to the system administrator or stopping the connection,.

In this scheme a hexadecimal encoding scheme was proposed for the IP addresses in the chromosome. The fitness of a chromosome depends on its ability to differentiate between normal and anomalous connections correctly. The actual validity of a rule is examined by matching the historical data set comprised of connections marked as either anomalous or normal. On successful match between a rule and an anomalous behavior, a bonus is given to the current chromosome and a penalty is applied if the chromosome matches with any of the normal connection.

The outcome of a connection is calculated based on whether a field of the connection matches the pre-classified data set, and it is then multiplied by the weight of that field. The weight values are identified according to different fields in the connection record as reported by network sniffers. Therefore, all genes representing the destination IP address field have the same weight. For example, in most attacks destination IP address is the target of an intrusion while the source IP address is the originator of the Intrusion, hence the destination IP address is awarded the highest weight fol-

lowed by source IP address. Other parameters that are used include the destination port number, source port number, duration of connection, etc. The Matched value is a Boolean value and the calculation of outcome is depicted in equation 3.2

$$Outcome = \sum Matched * Weight_i \tag{3.2}$$

The absolute difference between the outcome of the chromosome and the actual suspicious level is computed as a Δ factor. The $suspicious_{level}$ is a threshold that indicates the degree to which two network connections are considered as a "match".

$$\Delta = |Outcome - suspicious_{level}| \tag{3.3}$$

On the occurrence of a mismatch, a penalty is computed using the Δ factor. The ranking in the equation is a measure of whether or not an intrusion is easy to identify.

$$penalty = \frac{\Delta * ranking}{100} \tag{3.4}$$

$$fitness = 1 - penalty \tag{3.5}$$

Li et al.[13] suggested the use of a large number of rules as there are so many network connections possible. As there is a possibility that a small set of rules will

not detect the entire range of anomalies.

Bankovic et al. [9, 14] proposed a serial combination of two genetic algorithm based intrusion detection systems that provides the advantage of being simple due to its low computational overhead. The first IDS is used as a simple linear classifier for anomaly detection that differentiates normal connections from possible attacks. Because this system is known to exhibit a high rate of false-positives, the authors implemented an additional system based on if-then rules that is trained to recognize normal connections. This filters the output of the first system to considerably reduce the number of false positives. The linear classifier is trained using incremental Genetic Algorithm (GA) where each chromosome in the population comprises of four genes. The first three chromosomes characterize the coefficients of the linear classifier and the fourth one signifies the threshold value. The rules system is also similarly trained using incremental GA, where each rule is represented by a 3-gene chromosome. However, the population chosen in this case is significantly lower as compared to the first stage of the system. The division of the entire setup into two different systems allows each of the systems to be trained independently of the other.

Folorunso et al. [15] proposed an intrusion detection system called Intrusion Detection - Self Organizing Migrating Genetic Algorithm (ID-SOMGA) which is based on the combination of two algorithms: optimization algorithm - Self Organizing Migrating Algorithm (SOMA) and the Genetic Algorithm (GA). The motivation

behind this integration of SOMA and GA was its ability to handle both low and high population sizes and its advanced exploration capabilities. ID-SOMGA is a rule based system with the goal to develop rules that detect only the anomalous connections. These rules are tested on historical connections and are used to filter new connections to find suspicious network traffic. On successful detection of an anomalous connection, the rule is assigned a bonus point and otherwise a penalty is given. Although, ID-SOMGA exhibits a slower detection rate than an IDS with GA, it has a very low false positive rate.

Gong et al. [16] proposed a genetic algorithm based ID approach which is composed of two modules where each works in a different stage: the training stage and the ID stage. The training stage uses GA in an offline environment to generate rules from historical data. In the ID stage, the incoming traffic connections are classified in the real-time environment using the rules generated in the training stage. The authors chose two subsets from the 1998 DARPA dataset. Each record of the database consists of 9 network features and one added manually. A record is identified either as a normal connection or as a network intrusion based on the record type.

The fitness function in Genetic Algorithm is used as a metric to select the fit individuals who would undergo crossover and mutation to create the next generation population. Islam et al. [17] recommended a fitness function that is based on the accuracy-existence-occurrence structure.

The accuracy factor (af) represents the accuracy of rules. If a rule is represented

as, if A then C and the size of the training dataset is N, then: $af = |A\ and\ C|/|A|$; $existence = |A|/N$; $occurrence = |A\ and\ C|/N$. $|A|$ stands for the number of records that only satisfy condition A. $|C|$ stands for the number of records that only satisfy consequent C. $|A\ and\ C|$ stands for the number of records that satisfy both conditions A and consequent C.

A higher accuracy factor points to a strong rule to detect anomaly whereas a higher value of existence indicates that the rule matches more in the dataset. The value of existence is only needed for the random selection function of GA while both accuracy factor and the occurrence are to be included in the fitness function. The authors propose the use of accuracy factor and occurrence in a weighted form. A highly secured network demands a higher weight for accuracy and a faster security implementation implies a higher weight for occurrence.

Chapter 4

Attacks Description

There are four classes of attacks found in NSL-KDD dataset which is the enhanced version of the KDD99 intrusion detection dataset. A complete description of the dataset and the attacks is provided in [18, 19]. Each of the classes and the corresponding attacks are described below.

4.1 Denial of Service

The dictionary meaning for denial of service is to repudiate what is intended for you. In the context of computer and network security, a denial of service attack repudiates legitimate users of the requested service. The requested service may be a web access, a file download, etc. In this type of attack, the attacker makes the requested server too busy by either making establishing parallel connections to the

server or slowing it down. In both the cases a legitimate server is denied of the requested service or it takes too long for the user to access the service.

The most common type of denial of service attack is when the attack floods the servers with too much information to handle. For example instead of requesting a web address from the web browser the attacker types a large number of backslashes (back attack) which is mostly used as escape sequence. When the web server receives this information it takes more time in processing this request and hence the other users find the server unavailable.

A number of different denial of service attacks are found in the NSL-KDD dataset. In the following subsections some of these attacks are described in detail.

4.1.1 Apache2

The "Apche2" is a kind of denial of service attack that targets Apache web servers. In this attack, a user request consists of a large number of http headers. In order to process these headers the server takes more time, slows down and even crashes in many situations. The number of headers used in the attack can be varied by attacker.

For an intrusion detection system to detect Apache2 successfully should look for an anomalous number of headers in a request. Typically a web page consists of about 10 to 30 object ion a page so a network connection request with hundred or more headers can be considered as malicious.

4.1.2 Back

The "Back" attack is a kind of denial of service attack that targets Apache web servers. In this attack, a large number of front-slashes ('/') are submitted to the web server as a request. This large number of front slashes slows down the server and makes it unavailable for other requests. This attack consumes a large number of CPU cycles of the server and slows down all the other system or user related activities. When this attack is stopped by the attacker the system/server automatically recovers.

An intrusion detection system looking to detect back attack should account for the number of front slashes in the request. If the number is higher than a threshold then the connection should be considered as an attack.

4.1.3 Land

The "Land" attack is a kind of denial of service attack which targets relatively old TCP/IP machines. In this attack, a spoofed or a crafted SYN packet with the same source and destination IP address is sent to the machine/server. Under this attack, the CPU of the target machine is heavily utilized by processing the spoofed connection which slows the system down and ultimately results in denial of service. Any intrusion detection system can detect a Land attack easily by examining the source and destination IP addresses in the packet. In the KDD dataset, this is

indicated by the land field which is set when both the source and destination IP addresses are the same in a connection or a packet [20, 21].

4.1.4 Mailbomb

The "Mailbomb" attack is a kind of denial of service attack in which an attacker overflows the target machine/server queue by sending a large number of messages. These messages add up in the server's queue and result in a possible system failure. For an intrusion detection system to successfully detect a mailbomb attack, it should monitor the number of messages being sent from or sent to a certain user in a short period of time. This calls for determining the thresholds being imposed by the mail exchange servers and will be different for different networks.

4.1.5 Neptune

The "Neptune" attack is a kind of denial of service attack which can affect most of the TCP/IP based systems. Neptune attack is also known as SYN Flood attack. All half open tcp connections are recorded in the "tcpd" server. This data structure contains information on all the pending connections. In this attack this data structure is overwhelmed by creating a lot of such connections until the target system is not able to accept new incoming connections. The table is cleared after the timeout interval. But for crafted IP packets which request new connections quicker than the target system can timeout, the table will never clear its content and runs out of

memory, reject newer incoming connections and eventually crash [22].

For an intrusion detection system to detect a Neptune attack successfully, it should check the simultaneous number of SYN packets destined for a certain machine (computer). The source of the connection might not be known. If the number of simultaneous packets is higher than a defined threshold then it can be termed as a Neptune attack.

4.1.6 Ping of Death (POD)

The "Ping of Death" (pod) is a kind of denial of service attack in which the attacker sends oversized IP packets. Different systems behave differently under this attack [23]. Some of the known effects include system rebooting, abrupt halting and instant rebooting. This is the simplest of the attacks to create, e.g. any one can create oversized ping packets by using the command $ping - l < packetsize >$.

Any intrusion detection system looking to detect ping of death attack has to monitor the size of ICMP packets. For example any ICMP packet larger than 64000 bytes is considered as ping of death attack.

4.1.7 Smurf

The "smurf" attack is a kind of denial of service attack in which the attacker uses ICMP echo requests with a spoofed source address. The replies of these ICMP echo requests cause the target machine to crash. This attack involves three entities, the

first is the attacker, the second is the intermediate machine which receives the echo request from the attacker and sends a reply. And the third one is the victim machine which receives the replies. The intermediate machine can also be the target machine [24].

For intrusion detection system to detect this attack successfully it should monitor if there is a large number of "echo replies" being sent to a particular machine. These replies may originate from a different machine as well. Another striking feature of this attack is that no echo request is being sent from the victim machine.

4.1.8 Teardrop

The "teardrop" attack is a kind of denial of service attack that targets those TCP/IP implementations which are not able to deal with overlapping IP fragments properly. A teardrop attack causes the target system to reboot. This problem was usually found in older TCP/IP based machines [25].

An intrusion detection system aiming to detect this attack should analyze two IP packets. One with the fragment offset of 0 having a payload of **N** with the **MF** flag set, and the second with the MF flag cleared and offset of greater than N and a payload size of less than **N**.

4.1.9 Udpstorm

A "Udpstorm" attack is a kind of denial of service attack which aims to reduce the system efficiency by slowing the target machine down or congesting the network. In this attack, the attacker sends a spoofed packet (from the echo port) to the victim machines. The victim machines respond to the echo request by assuming that the other machine requested it. This process continues in a loop until a super user or network administrator stops it. This endless loop creates a denial of service [26].
The are two methods to detect "udpstorm" attack efficiently. The first method to detect if the packet has originated outside the local network. The second is by monitoring, if a large number of packets are being sent between the echo port of the victim machines.

4.2 User to Root (U2R)

This is a class of network attack in which the intruder begins with a normal user account (privileges) of the target system and is somehow able to acquire super-user access and privileges. This can be done by launching a brute force or a dictionary based attack to acquire the super-user privileges. Sniffing the password over unsafe connection is also one of the methods to access the administrator (super-user) access. A buffer_overflow attack is the most common attack of this class where an insufficient buffer space leads to the execution of arbitrary commands on the target machine.

One of the ways to avoid U2R attacks is careful programming but it has been seen that some utilities are susceptible to these kinds of attacks. In the following text different number of U2R attacks are explained.

4.2.1 Loadmodule

The "Loadmodule" attack is a kind of a U2R attack which exploits a bug in the loadmodule program in its ability to clear the current buffer and stack. By exploiting this vulnerability a user can acquire super-user privileges on the local machine.[27] A network based intrusion detection system can detect a loadmodule attack by observing the packet exchange between the attacker and the target machine. If the strings in the packet involve setting of configurable parameters of the system, the connection can be termed as anomalous.

4.2.2 Perl

The "Perl" attack is a kind of user to root attack which aims at exploiting flaws in perl implementations. Suidperl is a version of Perl that supports saved set-user-ID and set-group-ID scripts. In early versions of suidperl the interpreter does not properly relinquish its root privileges when changing its effective user and group IDs. On a system that has the suidperl, or sperl, program installed and supports saved set-user-ID and saved set-group-ID, anyone with access to an account on the system can gain root access.

4.3 Probe

This is a class of computer attacks in which an attacker scans the network connections and activities to find vulnerabilities and the network and the computers which are part of it [28]. Such type of activities are useful to intruders to launch attacks in future, if the vulnerabilities are not resolved. Satan, mscan, saint are among the most common probing tools and their ease of use does not require an attacker to be an expert, and can gain information of known vulnerabilities on target machines. The problem with this class of attacks is their time duration because an attacker can scan a machine frequently or slowly. Frequent scanning can be detected easily as once an attacker comes under the radar and if he is scanning machines in short time duration again and again it is easier for an intrusion detection system to detect it. But if an attacker is scanning one port a day, then it is quite difficult for any system to detect such kind of attacker. In the following text, some of the probing attacks in the NSL-KDD dataset are explained.

4.3.1 Ipsweep

The "Ipsweep" is a kind of probe attack in which an attacker sweeps the network to determine the active machines on the network. The attacker looks for the machines listening on the network. This information can then be used by attackers to launch attacks and exploit vulnerabilities. For an intrusion detection system to detect

ipsweep attack, it should monitor those ping requests which are destined to a number of computers on the network originating from the same source. The best way to determine an active machine on the network is through a ping test.

4.3.2 Mscan

Mscan is a kind of probing attack which utilizes domain name server zone transfers and exhaustive brute force attacks for scanning IP addresses to find machines with known vulnerabilities [29, 30].

A specific signature for mscan attack is hard to discover because it is dependent of the type of services or ports and the number of machines being scanned. Generally, an intrusion detection system can detect this attack by identifying connections from a single outside source to several ports on a single target system or to several machines in a short time period.

4.3.3 Nmap

Nmap is a widely available network scanning utility which is used to scan ports. The three most common scanning feature of this tool are UDP scanning, FIN scanning and SYN scanning of target machines. Nmap can be detected by an intrusion detection system by identifying a port-scan. A port scan can be related to TCP, UDP, FIN or SYN packets. A port-scan can be identified from connections from an external source to several ports on a single target system or to several machines in

a short time period [31].

4.3.4 Saint

Saint is an acronym for Security Administrator Integrated Network Tool which is used to probe information on information services and potential security flaws on network services such as finger, ftp, etc. Most of these flaws are operating system bugs, bugs in network utilities, incorrectly configured network services and bad policy decisions [32].

For an intrusion detection system to detect these attacks successfully, it should identify the distinct set network traffic which is created by scans. It has been found that in saint, almost all the scans performed are similar. So it is quite easy to detect this probe.

4.3.5 Satan

Satan is a probing attack quite similar to saint in the way it operates. The detection pattern for satan is the same for all scans performed. The following are the vulnerabilities scanned by Satan [33]:

1. NFS export to un-privileged programs.

2. NFS export via portmapper.

3. NIS passwords file access.

4. REXD access.

5. tftp file access.

6. Remote shell access.

7. Unrestricted NFS export.

8. Unrestricted X Server access.

9. write-able ftp home directory.

10. Several Sendmail vulnerabilities.

11. Several ftp vulnerabilities.

4.4 Remote to Local (R2L)

Remote to Local (R2L) attacks are caused by unauthorized access from a remote machine. In these attacks, the aim of the attacker is to gain unauthorized access to the local account of the server. Most of these attacks attempt to exploit vulnerabilities and bugs in system utilities. Different versions of these attacks are found in the data set. These attacks are mostly platform specific. Warezmaster and warezclient attacks are the most common examples of this attack category. In warezmaster attack the the attacker uses an anonymous account to create a directory of illegal software of the server. Once the directory is created, users of the ftp server can

access the directory to download illegal software which is the warezclient attack. Two of the R2L attacks are described below

4.4.1 ftp_write

The "ftp_write" attack is a kind of R2L attack which exploits flaws in ftp server configurations. If the anonymous ftp capabilities or directories are not configured as per the network policies, e.g. if root ftp directories are part of the same group in which ftp accounts are and if these directories are writable. Then it is possible for an attacker to add, modify, delete files and make an even acquire access to the local system [34].

In order to keep the network safe from ftp write attacks, the ftp server configuration should be correct. But for an intrusion detection system to detect this attack successfully, it should continuously monitor all the activities on the ftp servers. If any kind of file is created or modified in the root directory from an anonymous source it should be detected as ftp_write attack.

4.4.2 Phf

The "phf" attack is an R2L attack which exploits a poorly written CGI script and the attacker is able to complete his actions on the http server with the super-user privilege level. This attack was targeted at acquiring the httpd server's /etc/passwd file which contains all the passwords of the users in the system. In this attack the

attack does not have any account on the target machine.[35]

For an intrusion detection system to successfully detect this attack it should continuously monitor the http requests. If a request contains phf invocations and attempt to run some commands then it should detect as a “phf” attack [36].

Chapter 5

Network Intrusion Detection System Architecture

The proposed network intrusion detection system is composed of the following five modules.

1. Data Preprocessing Module.
2. Attack Modelling Module.
3. Anomalous Classification Module.
4. Fitness Evaluator
5. Evolutionary Module.

In the following text all the sections are described in detail of their working and implementation.

5.1 Data Preprocessing Module

This is the first block of the intrusion detection system. The main purpose of this block is to create attack masks. An attack mask is a binary string which indicates the features selected for detection of a particular attack. Attack mask comprises of 42 comma separated elements ordered with the KDD data set. A "1"in the attack mask string indicates that the feature is relevant and has been selected by the feature selection algorithm where as a "0"indicates that the field is don't care and is not relevant in the detection of that attack. The relevance of a field is determined by the feature selection algorithm automatically or is manually described in the attack chromosome modelling. The attack mask is used to create the attack chromosomes from the training set which serve as comparison strings for detection of attack from the test set.

Attack masks are created using the data mining software WEKA 3.6 [37] for feature ranking and subset evaluating simulations and are generated manually for the reverse engineering simulation.

For example, let us consider the following attack mask string for land attack generated using the chi-square feature ranking method:

0,1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0, 0,0,1,1,1,0,0,1,1,1,1,1,0,0,1

The string indicates that features 7(land), 38(dst_host_serror_rate), 39(dst_host_srv_serror_rate), 26 (srv_serror_rate), 4 (flag), 25 (serror_rate), 3 (service), 37 (dst_host_srv_diff_host_rate), 32 (dst_host_count), 5 (src_bytes), 36(dst_host_same_src_port_rate), 33 (dst_host_srv_count), 31(srv_diff_host_rate), 6(dst_bytes), 24(srv_count), 12(logged_in), 23(count), 2(protocol_type), 22(is_guest_login) and 21(is_host_login) are considered relevant in the detection by the chi-square algorithm. The rest of the features are don't care fields and will not be used in building the attack chromosomes.

It should be noted that the last feature is 1 and will remain 1 for all the attack masks and it indicates that the type of the connection will be taken as it is.

5.2 Attack Modelling Module

The "Attack Modelling Module"takes attack masks as input and outputs the attack chromosomes. Attack chromosomes comprise of 42 comma separated elements ordered with the KDD data-set with the last element indicating the attack type (class of data row). A "1"in the attack mask is replaced by the **value** of the corresponding field of the malicious connection and a "0"is replaced by an **X** which indicates that the field is a 'don't care'field.

Two classes of chromosomes can be created using the attack chromosome creation block

1. Pattern based Chromosome

2. Threshold based chromosome

5.2.1 Pattern based Chromosome

Pattern matching chromosomes detect attacks which exactly match with them. In other words these chromosomes are the exact signature of the attacks created from the training set and the connections which exactly match with the features and their values in the chromosomes are classified as attacks of the indicated type.

The complete training set is analyzed based on the attack mask provided and the sample size. Sample size is a predefined simulation parameter and accounts for the number of instances that will be analyzed in one run of the chromosome development step. For the pattern matching chromosomes all the connection with the distinct values are selected and added to the attack chromosome database. Table 5.1 is an example of pattern based chromosomes with the sample size of 10. As the pattern based chromosomes are based on exact matches so all the connections with distinct values as indicated by the attack mask will be selected for chromosomes. It can be seen that out of the ten sample connections selected, only one chromosome is constructed out of a number of similar connections and even a single different field causes a new chromosome to be added to the set of chromosomes generated.

5.2.2 Threshold based Chromosome

Threshold based chromosomes detect attacks which exceed the threshold defined in chromosome. For these chromosomes to detect attacks successfully the fields of protocol, service and flag should match exactly as they are part of the connection describing fields and are symbolic values so threshold value for them cannot be defined. The rest of the features including source and destination bytes are either continuous or binary values for which threshold can be defined. Based on the sample size provided, exact values for fields of protocol, service and flag are selected whereas for the rest of the fields the lower threshold is selected. All the connection is the test set which match exactly to the first four fields and exceed the threshold for the rest are classified as attacks of the indicated types.

Table 5.2is an example of threshold based chromosomes with the sample size of 10. The threshold for a particular field is created based on the floor of the values for that field. As the first four fields will be pattern based even in the threshold based values for the rest of the values lowest values if selected for the chromosome. The number of chromosomes using this technique is smaller than that in the pattern matching chromosomes.

Attack Mask
1 1 1 1 1 1 1 0 0 1 0 1 1 0 0 1 0 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Sample Connections Set
0 tcp http SF 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 3 3 0 0 0 0 1 0 0 109 109 1 0 0.01 0 0 0 0.01 0.01 back
0 tcp http RSTR 33336 2920 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 5 5 0 0 0.4 0.4 1 0 0 255 255 1 0 0 0 0 0 0.06 0.06 back
0 tcp http SF 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 6 7 0 0 0 0.14 1 0 0.29 255 255 1 0 0 0 0 0 0.05 0.05 back
0 tcp http RSTR 54060 2920 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 4 4 0 0 0.25 0.25 1 0 0 255 255 1 0 0 0 0 0 0.05 0.05 back
0 tcp http SF 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 1 0 0 39 39 1 0 0.03 0 0 0 0 0 back
0 tcp http SF 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 1 0 0 57 57 1 0 0.02 0 0 0 0.02 0.02 back
0 tcp http S2 54540 8315 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 6 6 0.17 0.17 0 0 1 0 0 255 255 1 0 0 0 0.01 0.01 0.04 0.04 back
0 tcp http RSTR 54540 7300 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0.5 0.5 1 0 0 255 255 1 0 0 0 0 0 0.05 0.05 back
0 tcp http SF 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 6 6 0 0 0.17 0.17 1 0 0 12 12 1 0 0.08 0 0 0 0.08 0.08 back
0 tcp http SF 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 3 3 0 0 0.33 0.33 1 0 0 96 96 1 0 0.01 0 0 0 0.16 0.16 back
Pattern based Chromosomes
0 tcp http SF 54540 8314 0 X X 2 X 1 1 X X X X X 0 X 0 0 3 3 0 X 0 0 1 0 0 109 109 1 0 0.01 0 0 0 0.01 0.01 back
0 tcp http RSTR 33336 2920 0 X X 1 X 1 0 X X X X X 0 X 0 0 5 5 0 X 0.4 0.4 1 0 0 255 255 1 0 0 0 0 0 0.06 0.06 back
0 tcp http S2 54540 8315 0 X X 1 X 1 0 X X X X X 0 X 0 0 6 6 0.17 X 0 0 1 0 0 255 255 1 0 0 0 0.01 0.01 0.04 0.04 back
0 tcp http RSTR 54540 7300 0 X X 1 X 1 0 X X X X X 0 X 0 0 2 2 0 X 0.5 0.5 1 0 0 255 255 1 0 0 0 0 0 0.05 0.05 back

Table 5.1: Pattern Based Chromosome example

Attack Mask
1 1 1 1 1 1 1 0 0 1 0 1 1 0 0 1 0 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Sample Connections Set
0 tcp http SF 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 3 3 0 0 0 0 1 0 0 109 109 1 0 0.01 0 0 0 0.01 0.01 back
0 tcp http RSTR 33336 2920 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 5 5 0 0 0.4 0.4 1 0 0 255 255 1 0 0 0 0 0 0.06 0.06 back
0 tcp http SF 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 6 7 0 0 0 0.14 1 0 0.29 255 255 1 0 0 0 0 0 0.05 0.05 back
0 tcp http RSTR 54060 2920 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 4 4 0 0 0.25 0.25 1 0 0 255 255 1 0 0 0 0 0 0.05 0.05 back
0 tcp http SF 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 1 0 0 39 39 1 0 0.03 0 0 0 0 0 back
0 tcp http SF 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 1 0 0 57 57 1 0 0.02 0 0 0 0.02 0.02 back
0 tcp http S2 54540 8315 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 6 6 0.17 0.17 0 0 1 0 0 255 255 1 0 0 0 0.01 0.01 0.04 0.04 back
0 tcp http RSTR 54540 7300 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0.5 0.5 1 0 0 255 255 1 0 0 0 0 0 0.05 0.05 back
0 tcp http SF 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 6 6 0 0 0.17 0.17 1 0 0 12 12 1 0 0.08 0 0 0 0.08 0.08 back
0 tcp http SF 54540 8314 0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 3 3 0 0 0.33 0.33 1 0 0 96 96 1 0 0.01 0 0 0 0.16 0.16 back
Threshold based Chromosomes
0 tcp http RSTR 33336 2920 0 X X 1 X 1 0 X X X X X 0 X 0 0 5 5 0 X 0.4 0.4 1 0 0 255 255 1 0 0 0 0 0 0.06 0.06 back
0 tcp http RSTR 54540 7300 0 X X 1 X 1 0 X X X X X 0 X 0 0 2 2 0 X 0.5 0.5 1 0 0 255 255 1 0 0 0 0 0 0.05 0.05 back

Table 5.2: Threshold Based Chromosome example

5.3 Anomaly Classification Module

This is the block implemented as the search engine in the intrusion detection system. Attack Chromosomes are provided as input to this block and then the whole test set is searched based on the attack chromosomes provided. The objective of the search process is to identify malicious connections in the test set based on the attack chromosomes provided.

The searching procedure is based on the type of chromosomes provided as input. If pattern based chromosomes are provided then the search algorithm looks for the exact matches in the values of attack chromosome and if threshold based chromosomes are provided then the search algorithm looks for malicious connections that exceed the threshold values defined by the chromosomes. The don't care fields (X) in the attack chromosomes are not compared with the corresponding values in the test set.

The significance of the pattern based chromosome is that they are based on exact signature of attacks whereas threshold based chromosomes rely on nominal values for normal connections or the minimum values that an attack connection might have and the connections with values greater than those specified in the chromosome are tagged malicious.

The output of this block is the count of number of connections correctly detected as the attacks by the chromosome and the number of normal connections being classi-

fied as attacks. As the technique is a misuse detection system so all the connections that pass through the system are considered as normal connections.

5.4 Fitness Evaluation Module

Fitness of the attack chromosomes is evaluated in this module. Fitness is a term associated with genetic algorithm which defines how good a given chromosome is. In intrusion detection fitness is actually a measure of calculating the accuracy of detection for a chromosome. It accounts for the number of connections correctly classified as attacks and the number of normal connections incorrectly classified as attacks. In other words fitness is a measure of goodness minus badness of a chromosome. Mathematically

$$\frac{\alpha}{A} - \frac{\beta}{B} \tag{5.1}$$

In the above equation, α (alpha) is the number of connections correctly classified as attacks β (beta) is the number of normal connections classified incorrectly as attack **A** is the number of attacks of that type and **B** is the number of normal connections. The efficiency of this fitness function is that the fitness value always remains between +1 and -1 with +1 being the best chromosome which has detected all the attacks of that type and -1 being the worst chromosome which has not detected any attack but has detected all the normal connections as attacks. Typically, a chromosome with a positive fitness value is considered to be good because the percentage of detected

attacks is greater than the percentage of false alarms.

5.5 Evolutionary Module

Genetic operations like crossover and mutation are performed in the evolutionary module. Both genetic operators' crossover and mutations are used to evolve the chromosomes from the initial population to the subsequent generations. Before the genetic operations are performed all the chromosomes are sorted based on their fitness.

The chromosomes are selected based on roulette wheel selection for crossover. Chromosomes with high fitness values are more likely to go into crossover for generating offsprings in subsequent generations. In the roulette wheel method chromosomes with smaller fitness values also have a chance to participate for the next generation but their probability of being selected in smaller as compared to those with high fitness values. Another method is random selection of parent chromosomes for the crossover where the chromosomes are selected randomly and every chromosome has an equal chance of participating in the next generation. The new offspring generated through crossover has the properties of their parents. The percentage of properties being transferred from parents depends on the cut-point. A simple cut-catenate crossover is being implemented for the intrusion detection system. The idea for this crossover is to generate new chromosomes which are valid and detect different at-

tacks which inhibit properties from other attacks. For example parent1 and parent 2 crossover to generate offspring1 and offspring2. The process is depicted in figure 5.1.

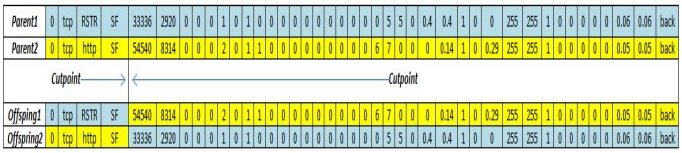

Figure 5.1: Crossover example

The second operator is mutation. The mutation operator introduces random changes in the chromosome which is useful to explore new possibilities of attacks in the test set. Mutation is implemented with the simple concept of bit flipping. For mutation a chromosome is selected at random and one of the 41 features within the chromosome is also randomly selected. Once the feature is selected it is replaced by another value of the same feature. Every feature is binary represented in the mutation block once the feature is selected its binary value is computed and one of the bits is inverted which is converted back to the string and replaced in the original chromosome.

For the values which are neither binary nor symbolic random changes are introduced. For example source and destination bytes are continuous values and are mutated by either incrementing or decrementing the original value.

5.6 System Flow

The modular description of the system is shown in figure 5.2 and the flow of data between them is illustrated in figure 5.3

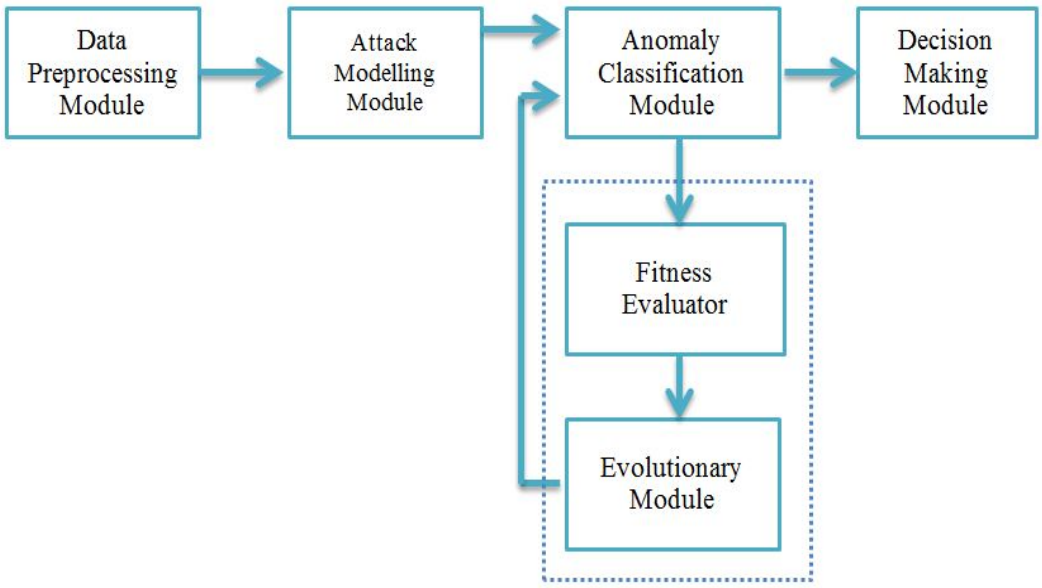

Figure 5.2: System Architecture

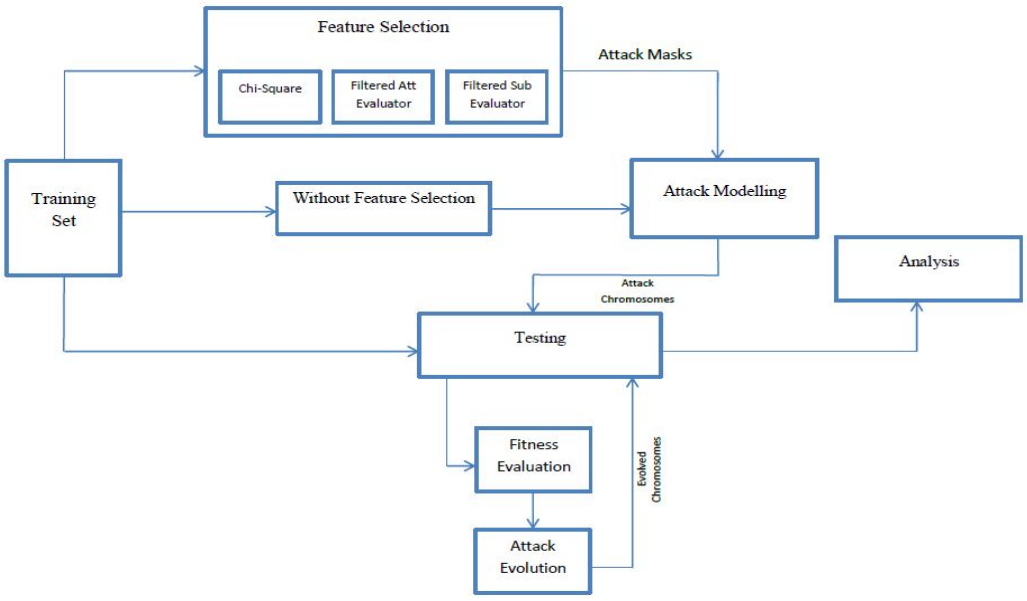

Figure 5.3: System Flow

Chapter 6

Intrusion detection with Feature Selection Techniques

6.1 Feature Selection

Feature selection is the process of determining features of the data set which are relevant for problem solving. In case of network intrusion detection (a problem of classifying dataset rows as per their type) feature selection refers to the problem of selecting those features which are relevant in the detection of that attack. The objective of feature selection is to determine a subset of features which is able to describe data as good as or better (if possible) than the original data set [38]. Feature Selection is important in those problems where it is either computationally infeasible to use all the features or where the estimation can be applied with limited data

samples. [39]

The advantage of feature selection is that it can be used to improve the performance of classification algorithms by using important and useful features only. Another advantage is that it enables the algorithm to scale up to the size of full feature set both in terms of space and time. It also helps is removing redundant and useless features.

Each connection is the KDD dataset is defined by 41 features. These 41 features include basic features which are obtained directly from packet header, content features acquired by applying domain knowledge on the data portion, time based traffic features which represent connection properties over a two second time interval and host based traffic features which contain traffic records based on the number of connections to the same host [40].

As each connection is described by 41 features it is quite difficult to develop a rule which successfully detects that attack [41]. It is also not feasible to manually test all the possible combinations of features that maximize the detection rate for that attack because it will call for the testing of 41! combinations ($3.34525266 \times 10^{49}$), to determine the best combination.

In order to achieve the best results several feature selection algorithms are employed to determine the best subset of features for detection of an attack. In this project I have employed four statistical methods including the attribute evaluator and subset evaluator based techniques to acquire the best set of features to maximize the

detection rate and reduce the false alarm rate.

6.2 Attribute Evaluation Methods

One of the methods for feature selection is the attribute evaluation method. The attribute evaluation method is some times called "Filter based methods ". These methods calculate the relevance of a feature with respect to the class and are independent of any classifiers [39]. The aim is to create a feature subset based on the most prominent features [42].

6.2.1 Chi-square

Chi-square is a statistical quantitative measure used to determine the relationship between two categorical variables. In the chi-square method first the frequencies are determined if no relationship exist between the two variables, second it quantifies the extent to which the observed distribution differs from the distribution determined in the first step. The chi-square test is used for ranking features as well.

The chi-square method is used in different ways [7, 43, 44] for intrusion detection. we have used this method to select the best set of features. All the 41 features of the KDD dataset are ranked to determine the best subset. The highest ranked feature indicates that the feature is the most relevant feature for the connection. The chi-square method to rank the features is given as follows:

1. A table is created with the frequencies of values for each attribute class.

2. Expected value of each attribute is calculated with the following formula:

$$EV = \frac{class\ count\ of\ the\ value * values\ count\ of\ that\ class}{total\ number\ of\ instances} \tag{6.1}$$

3. The chi-square value for that attribute is calculated as follows:

$$ChiVal\,(attribute) = \sum_{k=1}^{v} \sum_{m=1}^{c} ChiCell(expected_{k,l}) \tag{6.2}$$

4. Chicell is calculated as follows:

$$diff * \frac{diff}{expected} \tag{6.3}$$

Where $diff = expected - value\ frequency$

The ranked features based on chi-square are shown in table 6.1

Attacks	SN	Chi-Square Ranking
back	1	6, 5, 13, 10, 41, 40, 28, 27, 4, 24, 23, 37, 3, 35, 33, 34, 31, 12, 36, 2, 32, 38, 39, 1, 30, 29, 22, 16, 25, 19, 7, 21, 18, 17, 26, 20, 9, 11, 15, 8, 14
buffer overflow	2	10, 14, 13, 17, 1, 3, 5, 32, 33, 36, 6, 37, 24, 23, 35, 34, 31, 12, 4, 2, 22, 7, 21, 28, 29, 9, 8, 39, 40, 41, 38, 30, 20, 16, 19, 18, 11, 27, 15, 26, 25
ftp write	3	3, 9, 35, 32, 19, 17, 36, 22, 33, 23, 2, 4, 12, 7, 21, 10, 8, 37, 13, 11, 34, 40, 1, 41, 6, 38, 39, 5, 27, 29, 28, 26, 24, 25, 16, 14, 15, 30, 20, 31, 18
guess passwd	4	6, 5, 11, 40, 10, 4, 41, 39, 38, 3, 27, 28, 33, 32, 12, 36, 23, 24, 37, 35, 34, 31, 2, 22, 7, 21, 25, 20, 8, 30, 1, 26, 29, 14, 13, 16, 15, 19, 9, 17, 18

imap	5	3, 26, 4, 31, 28, 39, 24, 38, 25, 33, 5, 36, 32, 12, 37, 2, 22, 7, 21, 8, 30, 11, 9, 10, 40, 1, 41, 6, 35, 34, 19, 18, 23, 20, 27, 14, 29, 13, 17, 15, 16
ipsweep	6	37, 5, 3, 2, 32, 36, 31, 6, 35, 12, 33, 23, 24, 40, 34, 41, 28, 27, 4, 1, 38, 39, 30, 29, 26, 25, 10, 22, 16, 19, 13, 17, 14, 15, 11, 7, 21, 20, 8, 18, 9
land	7	7, 38, 39, 26, 4, 25, 3, 37, 32, 5, 36, 33, 31, 6, 24, 12, 23, 2, 22, 21, 29, 30, 10, 8, 9, 41, 1, 34, 40, 35, 18, 17, 27, 19, 20, 13, 11, 28, 16, 14, 15
loadmodule	8	18, 14, 17, 37, 3, 32, 36, 10, 33, 2, 12, 4, 22, 7, 21, 35, 8, 9, 13, 11, 34, 40, 1, 41, 6, 38, 39, 5, 23, 28, 27, 26, 24, 25, 16, 31, 15, 19, 20, 30, 29
multihop	9	18, 17, 16, 13, 14, 33, 10, 3, 36, 32, 22, 23, 2, 12, 4, 7, 21, 9, 38, 8, 35, 37, 11, 40, 1, 41, 6, 39, 5, 27, 19, 20, 24, 26, 25, 15, 34, 31, 28, 30, 29
neptune	10	30, 29, 4, 5, 3, 35, 34, 39, 38, 26, 25, 23, 33, 6, 12, 36, 32, 37, 24, 31, 41, 2, 40, 27, 28, 1, 10, 22, 16, 13, 19, 17, 14, 15, 11, 18, 7, 21, 20, 8, 9
nmap	11	5, 3, 37, 2, 36, 4, 39, 32, 6, 35, 31, 26, 38, 25, 33, 12, 23, 24, 34, 1, 40, 41, 28, 30, 27, 29, 10, 22, 16, 13, 19, 17, 14, 7, 21, 18, 20, 9, 8, 15, 11
perl	12	16, 18, 17, 14, 3, 33, 34, 12, 2, 4, 22, 7, 21, 10, 8, 9, 13, 36, 11, 37, 40, 1, 41, 6, 38, 39, 5, 24, 29, 23, 27, 28, 25, 26, 32, 15, 35, 20, 30, 31, 19
phf	13	10, 14, 19, 3, 12, 2, 4, 22, 7, 21, 11, 9, 37, 35, 15, 36, 13, 40, 39, 1, 41, 38, 8, 5, 6, 26, 27, 24, 25, 30, 31, 28, 29, 18, 34, 16, 17, 32, 23, 20, 33
pod	14	8, 5, 3, 37, 2, 33, 31, 36, 38, 32, 6, 35, 12, 24, 34, 23, 40, 1, 41, 39, 30, 29, 4, 28, 27, 26, 25, 22, 7, 21, 17, 16, 20, 18, 19, 10, 13, 11, 15, 9, 14
portsweep	15	4, 41, 3, 5, 34, 27, 33, 35, 28, 40, 1, 36, 6, 30, 29, 12, 23, 24, 37, 25, 32, 39, 31, 38, 2, 26, 10, 22, 16, 13, 19, 17, 14, 15, 7, 21, 18, 20, 8, 11, 9
rootkit	16	9, 16, 17, 3, 13, 14, 33, 11, 34, 1, 24, 23, 12, 2, 4, 22, 7, 21, 8, 37, 35, 36, 10, 39, 41, 40, 6, 38, 5, 20, 18, 19, 25, 27, 26, 30, 32, 31, 28, 15, 29
satan	17	29, 5, 35, 30, 27, 3, 40, 34, 25, 33, 38, 23, 41 , 4, 6, 28, 12, 32, 36, 37, 24, 31, 39, 2, 26, 1, 10, 22, 16, 19, 13, 17, 14, 15, 7, 21, 18, 20, 8, 11, 9
smurf	18	5, 3, 2, 24, 23, 36, 6, 12, 35, 37, 32, 40, 31, 33, 34, 38, 1, 41, 39, 30, 29, 4, 28, 27, 26, 25, 10, 22, 16, 13, 19, 17, 14, 15, 7, 21, 18, 20, 8, 11, 9
spy	19	38, 39, 18, 15, 3, 2, 12, 4, 22, 7, 21, 11, 9, 10, 14, 34, 35, 13, 40, 37, 1, 41, 36, 8, 5, 6, 26, 24, 25, 29, 30, 27, 28, 33, 19, 16, 17, 31, 23, 20, 32
teardrop	20	8, 5, 3, 24, 38, 25, 40, 23, 30, 36, 29, 27, 34, 6, 35, 2, 33, 12, 37, 32, 31, 1, 41, 39, 4, 28, 26, 10, 22, 16, 19, 7, 21, 17, 20, 18, 11, 9, 15, 13, 14
warez client	21	5, 10, 6, 3, 1, 36, 22, 37, 33, 32, 24, 23, 35, 40, 38, 39, 31, 34, 12, 2, 41, 30, 4, 28, 29, 27, 25, 26, 16, 19, 7, 21, 17, 15, 20, 18, 9, 11, 14, 8, 13

warezmaster	22	6, 1, 36, 5, 3, 33, 32, 24, 12, 23, 37, 22, 31, 2, 4, 7, 21, 9, 30, 8, 29, 10, 11, 39, 41, 40, 34, 38, 35, 28, 19, 20, 25, 27, 26, 15, 13, 14, 18, 16, 17

Table 6.1: Feature Ranking with Chi-Square

6.2.2 Filtered Attribute Evaluation Method

The information gain of an attribute "x"with respect to the class attribute "y"is the reduction in uncertainty about the class Y when the value of x is known. Information gain attribute evaluator is used to evaluate the features of the KDD training set. It evaluates the worth of a feature or attribute based on the information gain with respect to the class. The information gain ranks the attribute based on the separating classes of the training samples. The rank of an attributed is calculated using equation 6.4

$$InformationGain = D_x - D_{-x} \tag{6.4}$$

where D_x is the information which includes attribute x and D_x is the information which excludes the attribute x. The information is calculated using the entropy equation 6.5:

$$Entropy = \sum_{i=1}^{j} p_i \ log \ p_i \tag{6.5}$$

The ranked features based on filtered attribute evaluator are shown in table 6.2

Attacks	SN	FAE Ranking

back	1	6, 5, 13, 10, 41, 40, 28, 27, 4, 24, 23, 37, 3, 35, 33, 34, 31, 12, 36, 2, 32, 38, 39, 1, 30, 29, 22, 16, 25, 19, 7, 21, 18, 17, 26, 20, 9, 11, 15, 8, 14
buffer overflow	2	10, 14, 13, 17, 1, 3, 5, 32, 33, 36, 6, 37, 24, 23, 35, 34, 31, 12, 4, 2, 22, 7, 21, 28, 29, 9, 8, 39, 40, 41, 38, 30, 20, 16, 19, 18, 11, 27, 15, 26, 25
ftp write	3	3, 9, 35, 32, 19, 17, 36, 22, 33, 23, 2, 4, 12, 7, 21, 10, 8, 37, 13, 11, 34, 40, 1, 41, 6, 38, 39, 5, 27, 29, 28, 26, 24, 25, 16, 14, 15, 30, 20, 31, 18
guess passwd	4	6, 5, 11, 40, 10, 4, 41, 39, 38, 3, 27, 28, 33, 32, 12, 36, 23, 24, 37, 35, 34, 31, 2, 22, 7, 21, 25, 20, 8, 30, 1, 26, 29, 14, 13, 16, 15, 19, 9, 17, 18
imap	5	3, 26, 4, 31, 28, 39, 24, 38, 25, 33, 5, 36, 32, 12, 37, 2, 22, 7, 21, 8, 30, 11, 9, 10, 40, 1, 41, 6, 35, 34, 19, 18, 23, 20, 27, 14, 29, 13, 17, 15, 16
ipsweep	6	37, 5, 3, 2, 32, 36, 31, 6, 35, 12, 33, 23, 24, 40, 34, 41, 28, 27, 4, 1, 38, 39, 30, 29, 26, 25, 10, 22, 16, 19, 13, 17, 14, 15, 11, 7, 21, 20, 8, 18, 9
land	7	7, 38, 39, 26, 4, 25, 3, 37, 32, 5, 36, 33, 31, 6, 24, 12, 23, 2, 22, 21, 29, 30, 10, 8, 9, 41, 1, 34, 40, 35, 18, 17, 27, 19, 20, 13, 11, 28, 16, 14, 15
loadmodule	8	18, 14, 17, 37, 3, 32, 36, 10, 33, 2, 12, 4, 22, 7, 21, 35, 8, 9, 13, 11, 34, 40, 1, 41, 6, 38, 39, 5, 23, 28, 27, 26, 24, 25, 16, 31, 15, 19, 20, 30, 29
multihop	9	18, 17, 16, 13, 14, 33, 10, 3, 36, 32, 22, 23, 2, 12, 4, 7, 21, 9, 38, 8, 35, 37, 11, 40, 1, 41, 6, 39, 5, 27, 19, 20, 24, 26, 25, 15, 34, 31, 28, 30, 29
neptune	10	30, 29, 4, 5, 3, 35, 34, 39, 38, 26, 25, 23, 33, 6, 12, 36, 32, 37, 24, 31, 41, 2, 40, 27, 28, 1, 10, 22, 16, 13, 19, 17, 14, 15, 11, 18, 7, 21, 20, 8, 9
nmap	11	5, 3, 37, 2, 36, 4, 39, 32, 6, 35, 31, 26, 38, 25, 33, 12, 23, 24, 34, 1, 40, 41, 28, 30, 27, 29, 10, 22, 16, 13, 19, 17, 14, 7, 21, 18, 20, 9, 8, 15, 11
perl	12	16, 18, 17, 14, 3, 33, 34, 12, 2, 4, 22, 7, 21, 10, 8, 9, 13, 36, 11, 37, 40, 1, 41, 6, 38, 39, 5, 24, 29, 23, 27, 28, 25, 26, 32, 15, 35, 20, 30, 31, 19
phf	13	10, 14, 19, 3, 12, 2, 4, 22, 7, 21, 11, 9, 37, 35, 15, 36, 13, 40, 39, 1, 41, 38, 8, 5, 6, 26, 27, 24, 25, 30, 31, 28, 29, 18, 34, 16, 17, 32, 23, 20, 33
pod	14	8, 5, 3, 37, 2, 33, 31, 36, 38, 32, 6, 35, 12, 24, 34, 23, 40, 1, 41, 39, 30, 29, 4, 28, 27, 26, 25, 22, 7, 21, 17, 16, 20, 18, 19, 10, 13, 11, 15, 9, 14
portsweep	15	4, 41, 3, 5, 34, 27, 33, 35, 28, 40, 1, 36, 6, 30, 29, 12, 23, 24, 37, 25, 32, 39, 31, 38, 2, 26, 10, 22, 16, 13, 19, 17, 14, 15, 7, 21, 18, 20, 8, 11, 9
rootkit	16	9, 16, 17, 3, 13, 14, 33, 11, 34, 1, 24, 23, 12, 2, 4, 22, 7, 21, 8, 37, 35, 36, 10, 39, 41, 40, 6, 38, 5, 20, 18, 19, 25, 27, 26, 30, 32, 31, 28, 15, 29
satan	17	29, 5, 35, 30, 27, 3, 40, 34, 25, 33, 38, 23, 41 , 4, 6, 28, 12, 32, 36, 37, 24, 31, 39, 2, 26, 1, 10, 22, 16, 19, 13, 17, 14, 15, 7, 21, 18, 20, 8, 11, 9
smurf	18	5, 3, 2, 24, 23, 36, 6, 12, 35, 37, 32, 40, 31, 33, 34, 38, 1, 41, 39, 30, 29, 4, 28, 27, 26, 25, 10, 22, 16, 13, 19, 17, 14, 15, 7, 21, 18, 20, 8, 11, 9

spy	19	38, 39, 18, 15, 3, 2, 12, 4, 22, 7, 21, 11, 9, 10, 14, 34, 35, 13, 40, 37, 1, 41, 36, 8, 5, 6, 26, 24, 25, 29, 30, 27, 28, 33, 19, 16, 17, 31, 23, 20, 32
teardrop	20	8, 5, 3, 24, 38, 25, 40, 23, 30, 36, 29, 27, 34, 6, 35, 2, 33, 12, 37, 32, 31, 1, 41, 39, 4, 28, 26, 10, 22, 16, 19, 7, 21, 17, 20, 18, 11, 9, 15, 13, 14
warez client	21	5, 10, 6, 3, 1, 36, 22, 37, 33, 32, 24, 23, 35, 40, 38, 39, 31, 34, 12, 2, 41, 30, 4, 28, 29, 27, 25, 26, 16, 19, 7, 21, 17, 15, 20, 18, 9, 11, 14, 8, 13
warezmaster	22	6, 1, 36, 5, 3, 33, 32, 24, 12, 23, 37, 22, 31, 2, 4, 7, 21, 9, 30, 8, 29, 10, 11, 39, 41, 40, 34, 38, 35, 28, 19, 20, 25, 27, 26, 15, 13, 14, 18, 16, 17

Table 6.2: Feature Ranking with Filtered Attribute evaluator

6.2.3 Simulation and Results

All the attacks were individually treated and ranking methods were applied by comparing them with the normal connections. This was done to identify features which differentiate an attack from the normal connections. All the experiments for ranking the features were performed using the WEKA 3.6 data-mining tool. The attacks and the normal connection were provided to the tool in an "arff"file (file format for weka) and the tool provided the results.

For the results a technique proposed by Dr. Zubair Baig was employed to determine the effect of varying the number of features on the detection rate and false alarms rate. Another objective of this simulation was to determine the optimal number of features for detection of each attack. The simulation was carried out by decreasing

the number of features. The first simulation was done with all the important features indicated by the ranking algorithm. The importance of a features is indicated its value. Any feature with a non zero value is considered relevant. The number of features was reduced by 10% for every simulation and the effect was observed on the number of true positive and false positives obtained. True positive (TP) is the count of malicious connections correctly detected/classified and False positive (FP) is the count of normal connections classified as attacks.

The results in the figures 6.1 to figure 6.10 are plotted in terms of percentages in order to avoid the gap between the attacks with large number of connections and small number of connections. The results are plotted for all the 22 attacks where **TP** represents the percentage of true positives and **FP** shows the percentage of false positives. A complete analysis of these results is provided later in the section.

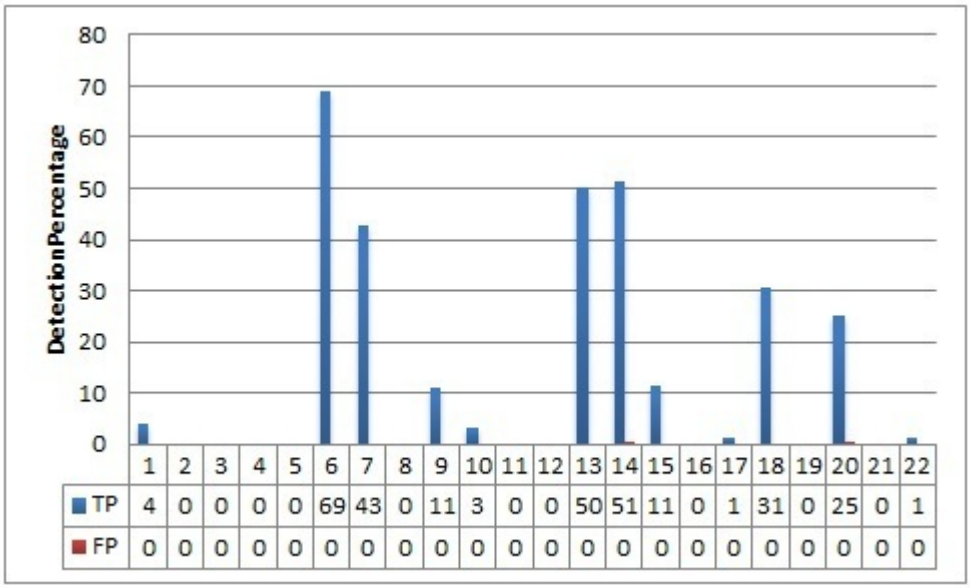

Figure 6.1: TP and FP rates with Full feature set

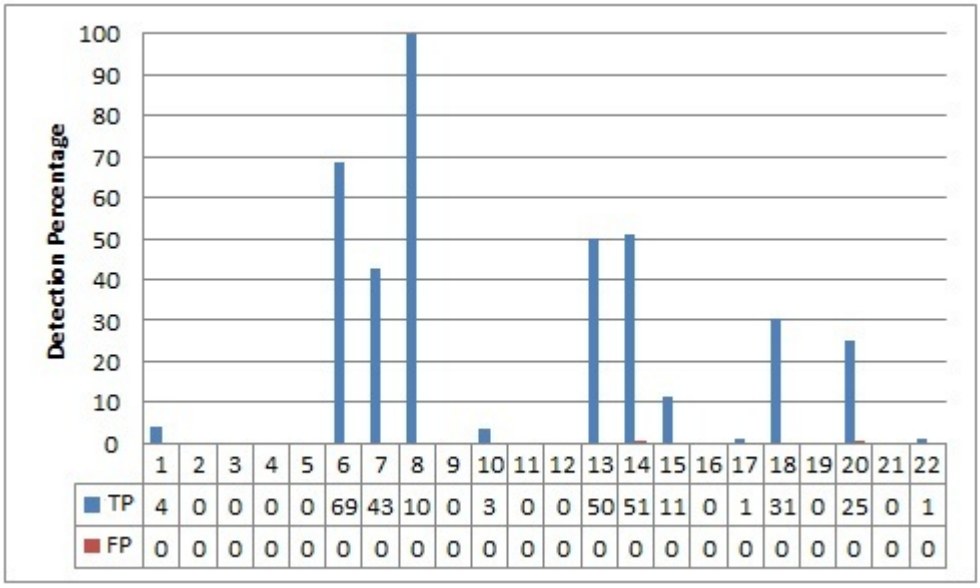

Figure 6.2: TP and FP rates with 90% feature set

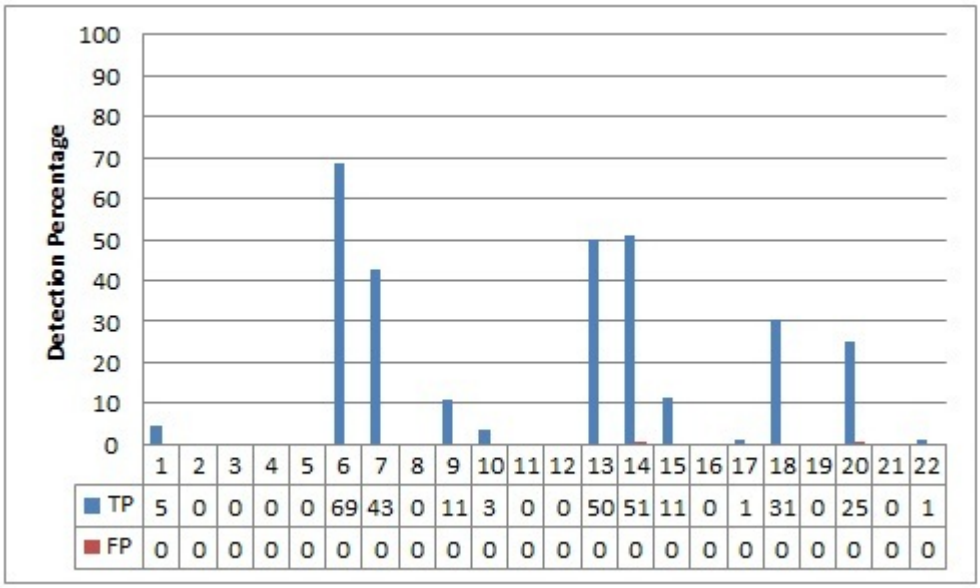

Figure 6.3: TP and FP rates with 80% feature set

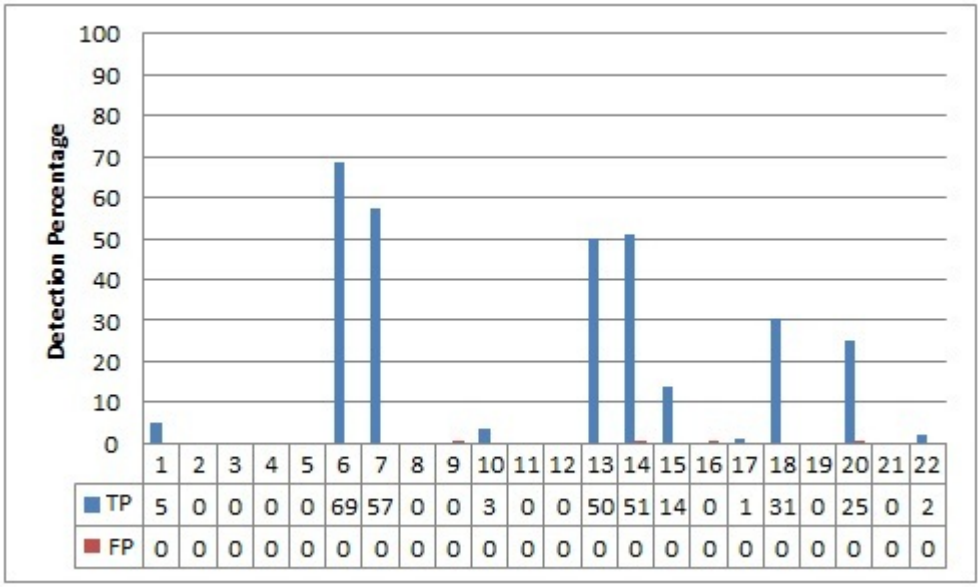

Figure 6.4: TP and FP rates with 70% feature set

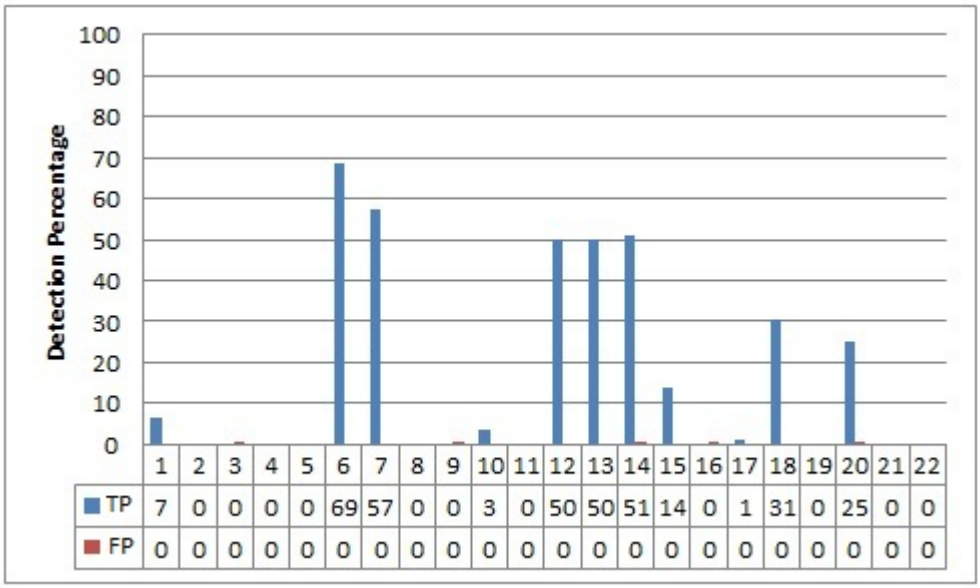

Figure 6.5: TP and FP rates with 60% feature set

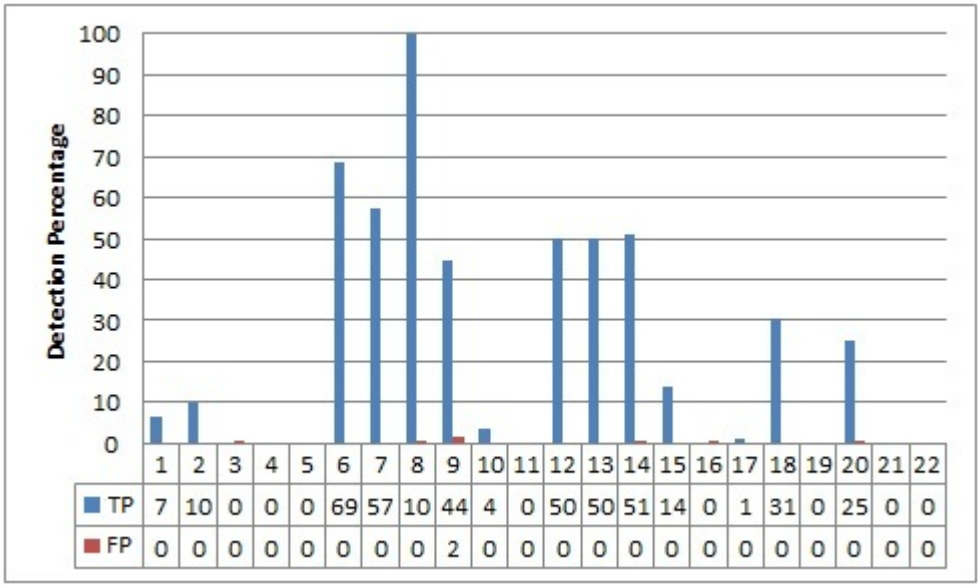

Figure 6.6: TP and FP rates with 50% feature set

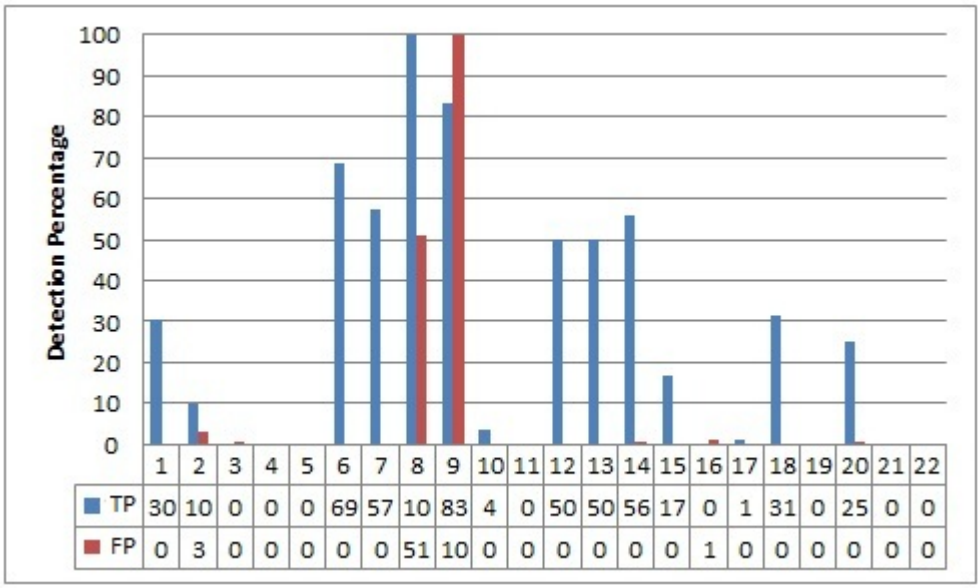

Figure 6.7: TP and FP rates with 40% feature set

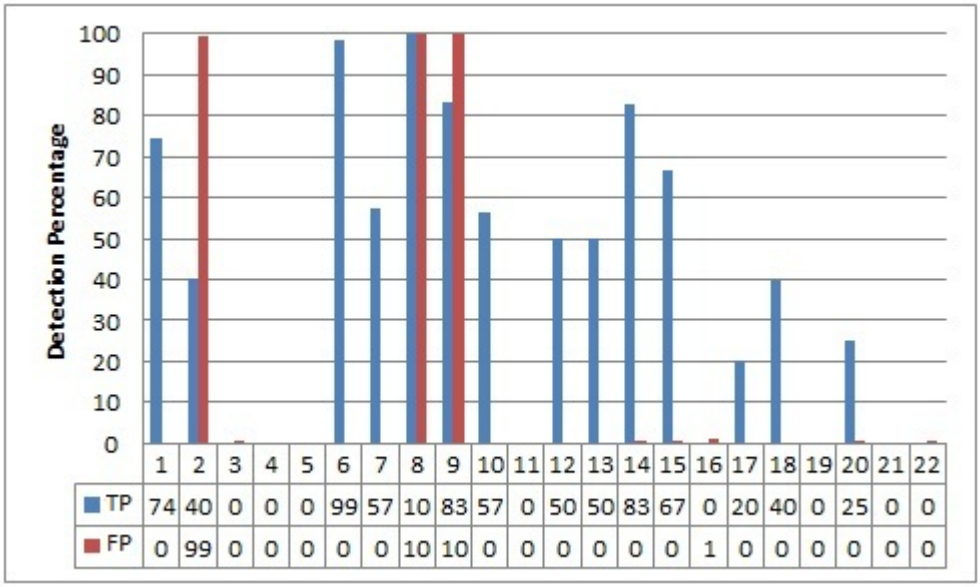

Figure 6.8: TP and FP rates with 30% feature set

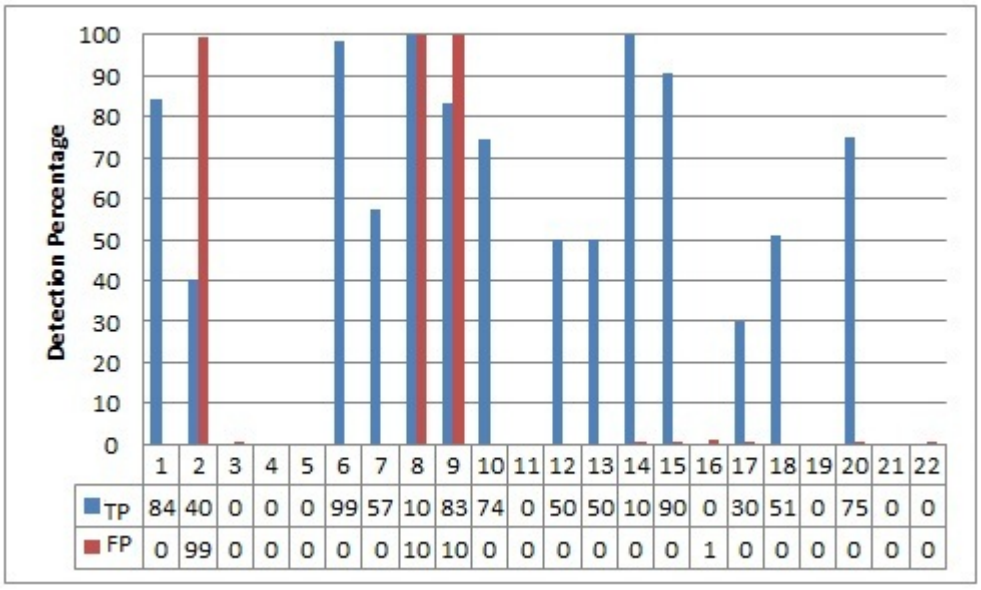

Figure 6.9: TP and FP rates with 20% feature set

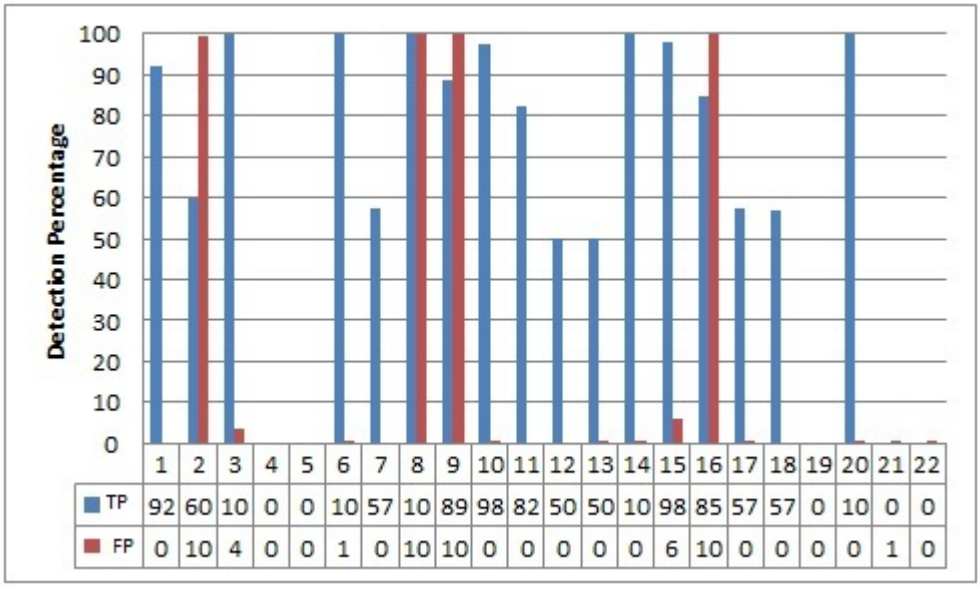

Figure 6.10: TP and FP rates with 10% feature set

6.2.4 Analysis

The results for the feature selection indicate that with a higher number of features, the true positive detection rate is low for most of the attacks and also the false positive percentage is low. But, as the number of features is decreased we see a remarkable increase in the true positive detection rate.

A similar trend is observed in the results with half of the features selected. As the number of features is reduced further beyond 50%, the number of false positives start to increase. The results with only 10% of the features selected show the maximum detection rate for attacks belonging to Denial of service and Probe classes of attacks.

The true positive detection rate is low throughout the simulation for U2R and R2L attacks mainly because of the difference in the samples of training and test set data. It can also be observed that different attacks show maximum detection rates with different number of features. For example loadmodule shows a maximum true positive detection rate of 100% with 90% of the feature subset selected, and with the feature set of less than 50% the TP detection rate is high but the FP detection is very high which is not a good sign for the efficiency of the intrusion detection system.

Similarly it can be seen that back and smurf attacks have better TP detection rates with lesser number of features. But with smaller number of features the FP rate is

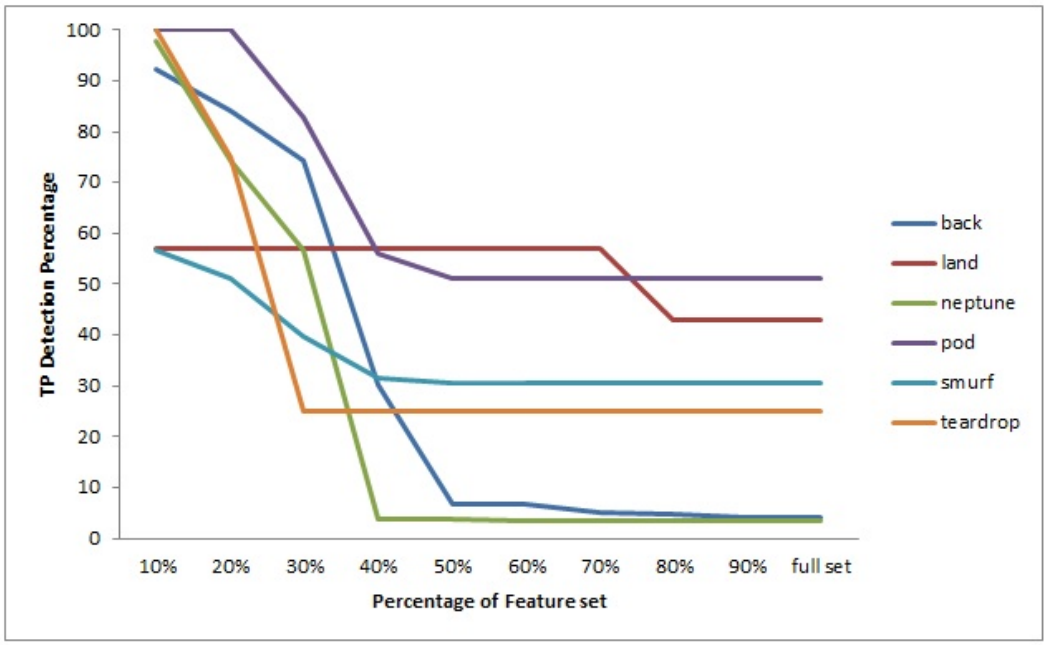

Figure 6.11: Comparison of TP rates for DoS attacks using Ranking method

very high for U2R class attacks. For R2L attacks not a single connection is detected based on the features selected using feature ranking methods.

It is inferred from the simulation that different type of attacks have good TP detection rates with different number of features.

A comparison of TP for the DoS, probe and U2R attacks is shown in figure 6.11, 6.12 and 6.13 respectively. Whereas the FP comparison for DoS, probe and U2R attacks is shown in figure 6.14, 6.15, 6.16 and 6.17 respectively.

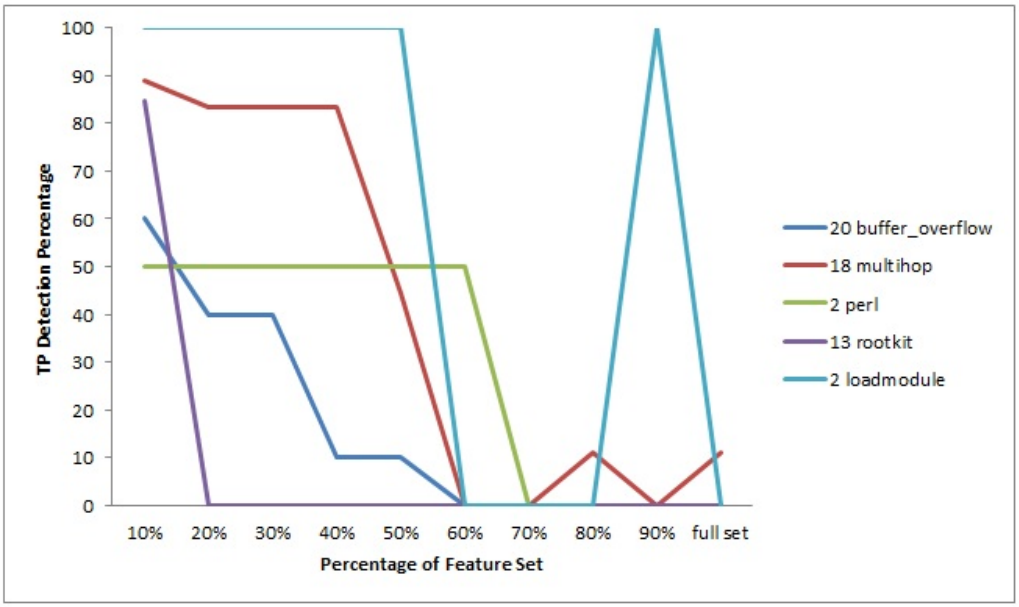

Figure 6.12: Comparison of U2R TP percentage using Ranking method

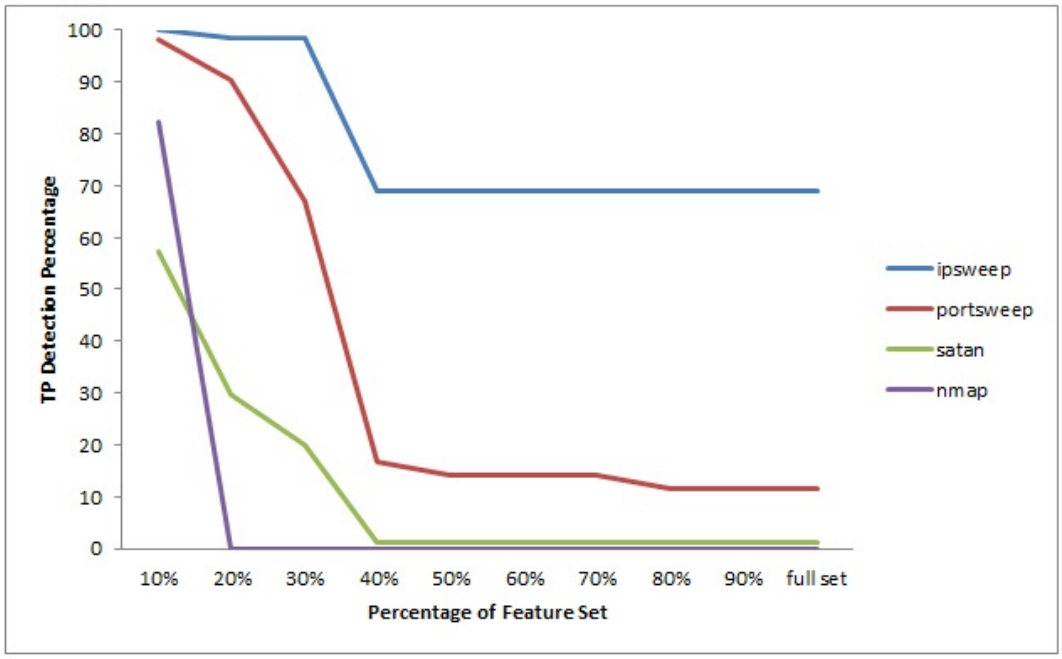

Figure 6.13: Comparison of Probe TP percentage using Ranking method

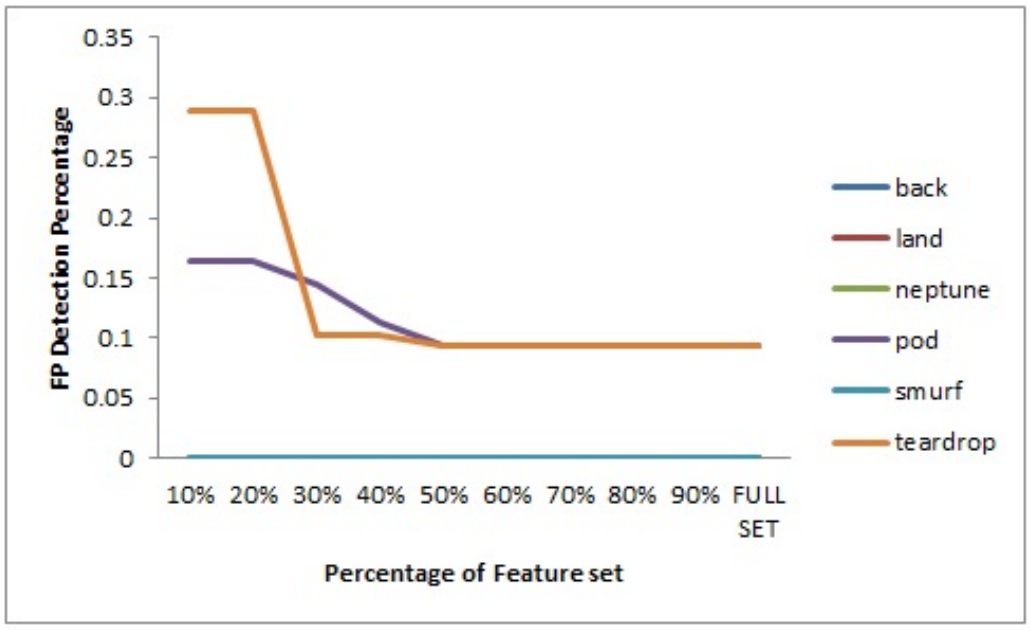

Figure 6.14: Comparison of DoS FP percentage using Ranking method

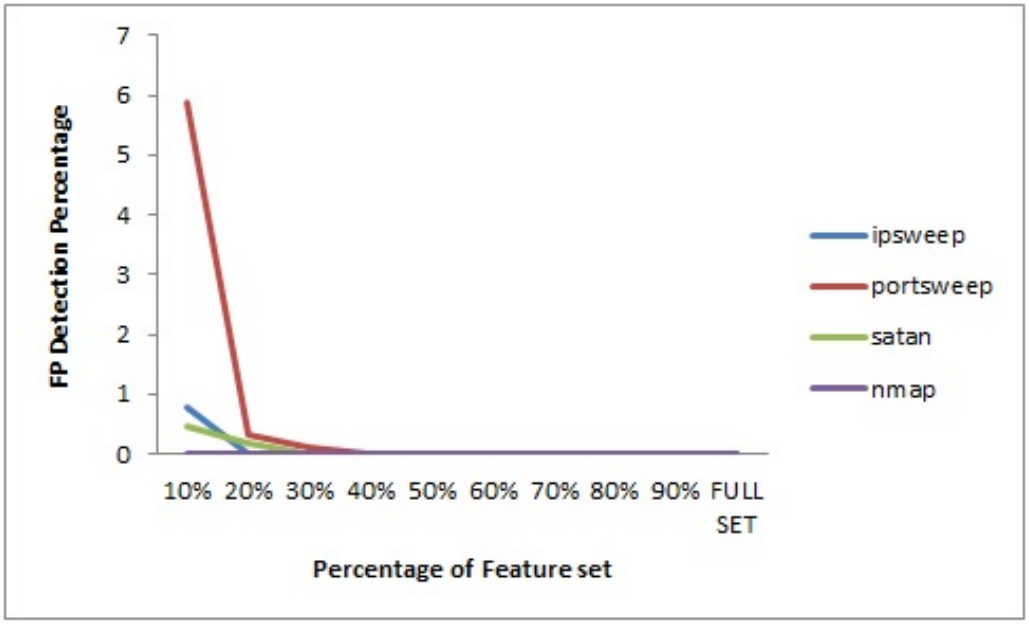

Figure 6.15: Comparison of Probe FP percentage using Ranking method

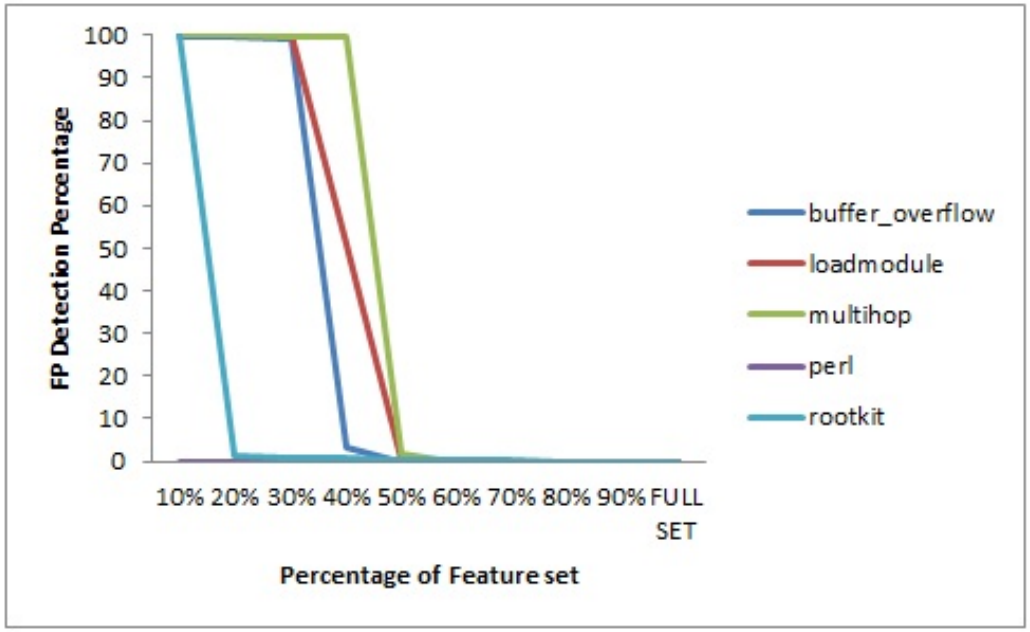

Figure 6.16: Comparison of U2R TP percentage using Ranking method

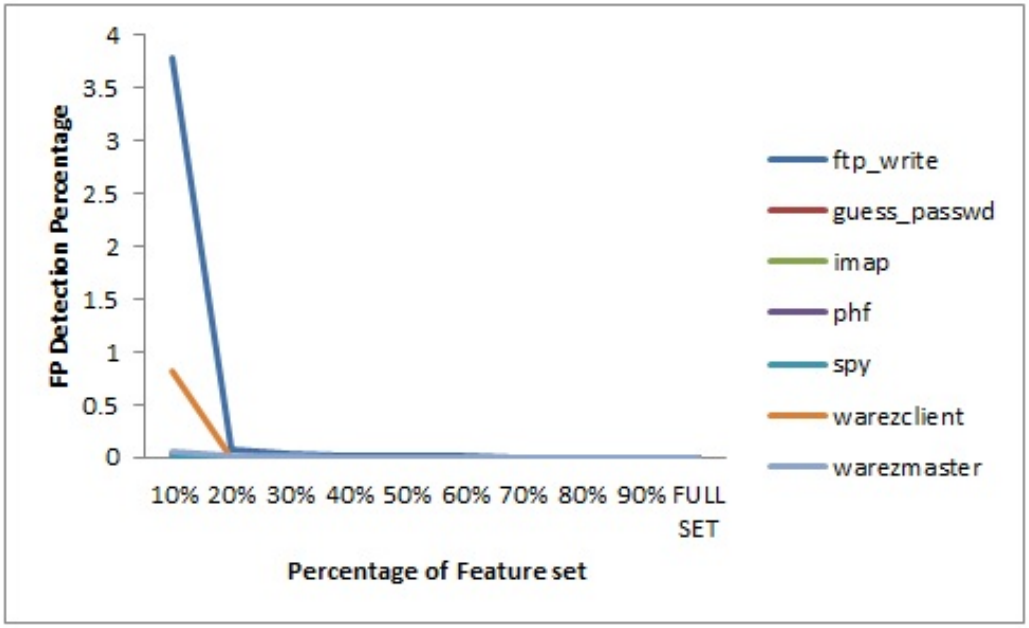

Figure 6.17: Comparison of R2L FP percentage using Ranking method

6.3 Subset Evaluation Methods

As compared to attribute evaluation method, subset evaluation methods calculate the relevance of a subset of features with respect to the class. The main aim of these methods is to create a feature subset that replaces the complete feature set for that class.

6.3.1 Filtered Subset Evaluator

The filtered subset evaluator works on the same principle as filtered attribute evaluator but the difference is that it selects a number of attributes to form the subset of features and then evaluates their efficiency in predicting the class of the connection.

6.3.1.1 Simulation and Results

The complete training data-set was evaluated by the Filtered subset Evaluator using WEKA 3.6 data mining tool. Two simulations were performed. In the first simulation all the attacks were individually compared with the normal connection in order to determine the subset of features which differentiate them from normal connections. The selected subset of features for every attack by comparing it with normal connection is shown in table 6.3

SN	Attack	Differentiating features
1	back	5, 10, 13
2	buffer overflow	14
3	ftp write	9

4	guess passwd	11
5	imap	26, 39
6	ipsweep	5
7	land	7
8	loadmodule	18
9	multihop	16,18
10	neptune	4,5,6,12,25,26,29,30,35,38,39
11	nmap	2,5,6,26,31,36,37,39
12	perl	18
13	phf	14
14	pod	8
15	portsweep	4,5,6,27,28,33,34,40,41
16	rootkit	9
17	satan	5,25,27,29,30
18	smurf	2, 5, 6
19	spy	15, 18, 39
20	teardrop	8
21	warezclient	10
22	warezmaster	6

Table 6.3: Selected feature subset with Filtered Subset Evaluator

The objective of second simulation was to identify features which distinguish different malicious connections from each other and from normal connections as well. In this simulation, all the attacks were simultaneously provided to the evaluator. Table 6.4 shows the feature subset by comparing the complete training data-set with the normal connections.

Attack	Selected features
Complete data-set	2, 3, 4, 5, 6, 8, 10, 12, 23, 25, 29, 30, 35, 36, 37, 38, 39, 40

Table 6.4: Selected feature subset for complete data-set using FSE

The results of the first simulation are summarized in table 6.5 and are plotted in

figure 6.18 and 6.19 where as the results of the second simulation are summarized in table 6.6 and the detection comparison is shown in fig 6.20 and 6.21

6.3.1.2 Analysis

The results of the simulation for filtered subset evaluator feature selection method show good TP detection for Denial of service and probe class of attacks. The percentage of false positives is also low for these classes. For U2R and R2L classes, the TP detection percentage is good but the FP percentage is mostly 100%. It means that the rules generated using are detecting a good percentage of attacks but they are classifying all the normal connections as attacks, which by no means is an acceptable solution.

By analyzing the feature subset for DoS and Probe classes it is clear that a number of features are selected for each of the attacks. On the other hand for U2R and R2L classes, for most of the attacks a single feature is selected. This supports the argument that we made in section 6.2.4 about these classes.

For the second simulation with the complete dataset no particular class show a remarkable TP detection percentage with the exception of ipsweep and imap.

From the results, it can be said that TP detection rate can be maximized by using a minimum number of correctly selected features. The minimum number of features are different for every attack and the results also show that if only a single feature is selected, a good TP detection percentage is over shadowed by a worst FP detection

percentage.

To achieve the objective of maximizing the TP detection rate it is necessary to select features as per the class of every attack and a tradeoff should be made for selecting the features which maximizes the TP detection percentage and minimizes the FP detection percentage.

SN	Attack	TP	FP
1	back	94.42896936	0
2	buffer overflow	100	100
3	ftp write	100	100
4	guess passwd	100	99.9691072
5	imap	100	96.03542375
6	ipsweep	100	1.760889713
7	land	100	0
8	loadmodule	100	100
9	multihop	83.33333333	99.9279168
10	neptune	81.68348722	0
11	nmap	80.82191781	0
12	perl	100	0
13	phf	50	0.185356812
14	pod	100	99.61898878
15	portsweep	73.24840764	0.31922562
16	rootkit	92.30769231	100
17	satan	48.43537415	2.090412934
18	smurf	56.84210526	0
19	spy	0	0
20	teardrop	100	0.535475234
21	warezclient	0	99.90732159
22	warezmaster	0	0.051488003

Table 6.5: TP and FP percentage for Filtered Subset Evaluator Simulation 1

The results of the first simulation are summarized in table 6.5 and are plotted in figure 6.18 and 6.19 where as the results of the second simulation are summarized

in table 6.6 and the detection comparison is shown in fig 6.20 and 6.21

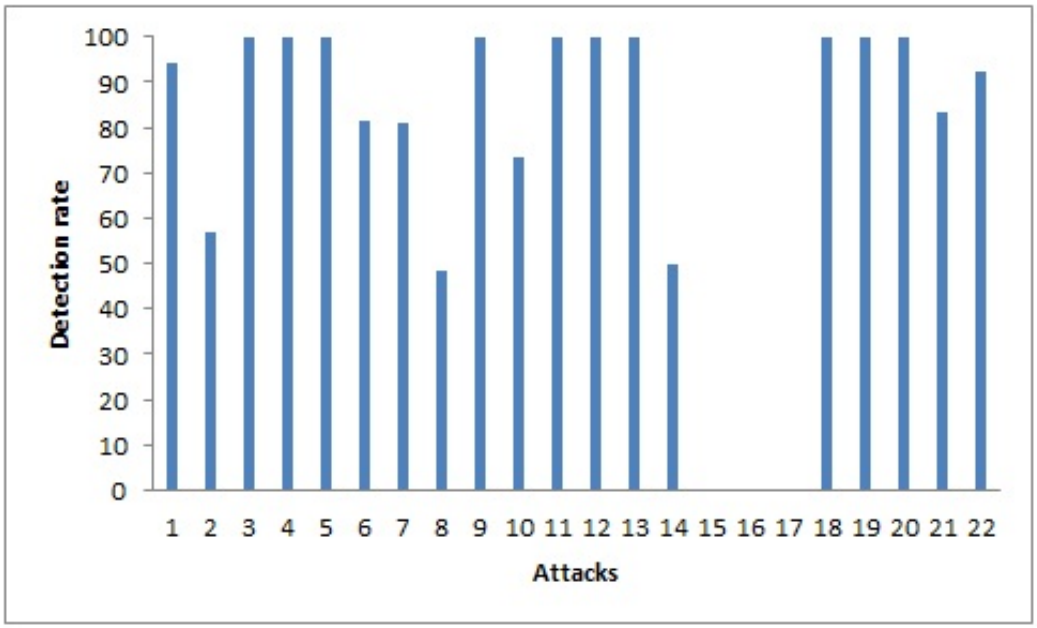

Figure 6.18: TP detection rate for entire dataset using FSE Simulation 1

SN	Attack	TP %	FP %
1	back	19.49860724	0
2	buffer overflow	0	0
3	ftp write	0	0
4	guess passwd	0	0
5	imap	98.58156028	0
6	ipsweep	100	1.760889713
7	land	57.14285714	0
8	loadmodule	0	0
9	multihop	0	0
10	neptune	11.89607043	0
11	nmap	2.739726027	0
12	perl	0	0
13	phf	0	0
14	pod	60.97560976	0.123571208
15	portsweep	15.92356688	0
16	rootkit	0	0
17	satan	1.224489796	0

18	smurf	0	0
19	spy	0	0
20	teardrop	25	0.092678406
21	warezclient	0	0
22	warezmaster	0	0

Table 6.6: TP and FP percentage for Filtered Subset Evaluator Simulation 2

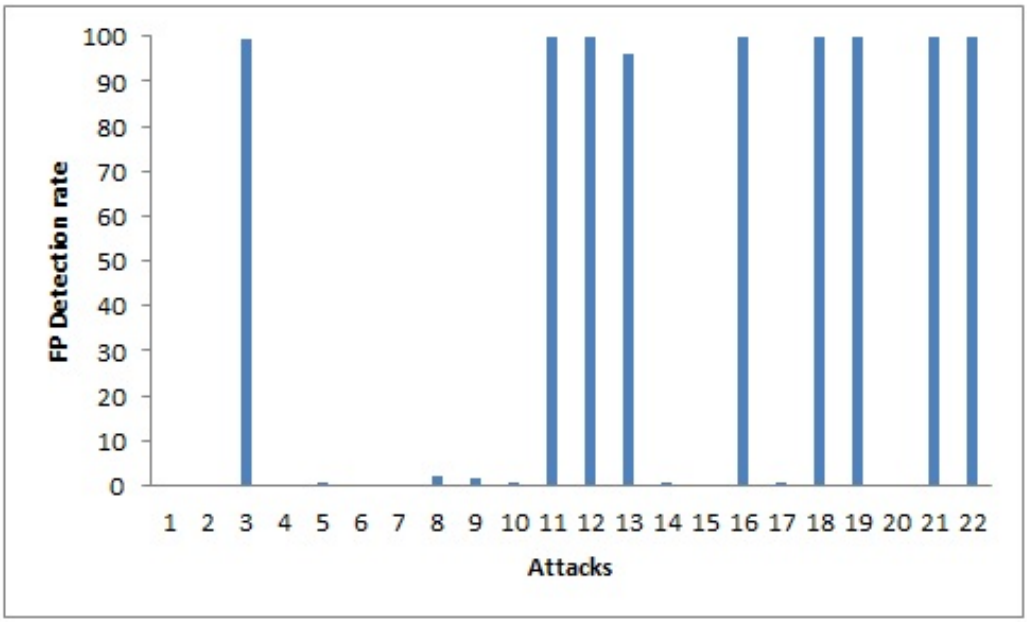

Figure 6.19: FP detection rate for entire dataset using FSE Simulation 1

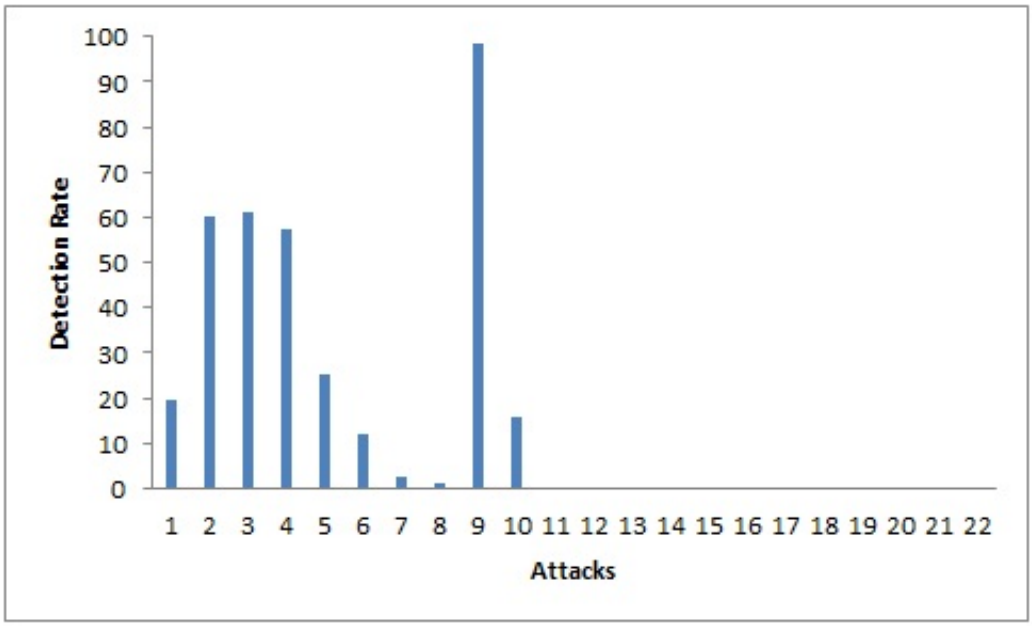

Figure 6.20: TP detection rate for entire dataset using FSE Simulation 2

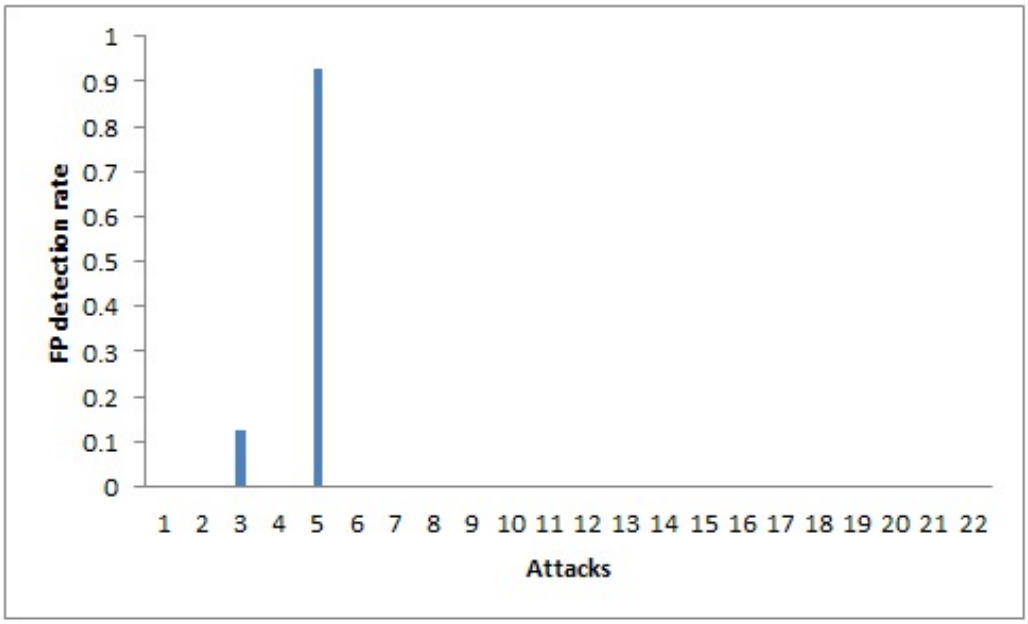

Figure 6.21: FP detection rate for entire dataset using FSE Simulation 2

Chapter 7

Chromosome Modelling for Intrusion Detection

7.1 Modelling of Malicious Code/Attacks

The internet today is the main cause of spreading of malicious software throughout the world. Viruses, worms, etc are the major forms of these malicious software. With millions of computers connected together and enormous amount of data exchanged between them these viruses spread very quickly. One of the solutions to stop spreading of these viruses and worms is through intrusion detection and prevention systems.

For any intrusion detection system to detect these attacks successfully it is required that the system should possess knowledge on detecting them. A number of such

techniques have been discussed in chapter 3.

Most of the intrusion detection systems are based on signature and rely of the matching of these signatures to detect intrusions. Reverse engineering can be applied in development of these signatures so that a higher detection rate can be achieved. In reverse engineering of these attacks, each and every step of the malicious software's activity can be tracked and the extent of damage can be studied and a mechanism can be develop to counter them. A complete description of methods for identifying vulnerabilities in a computer system and how reverse engineering can be applied to deal with these malicious softwares is provided in [45].

The proposed scheme is based on the modelling of intrusion-based attacks through genetic chromosomes, and their subsequent evolution for detecting both known as well as unknown malicious attacks. The attacks are modelled as chromosomes through feature selection and ranking methods, namely, Chi-square and Filtered attribute evaluation and Filtered Subset evaluation techniques. These technique use statistical analysis to determine a subset of features with the best predictive ability. Finally the best subset of features is selected by comparing attack signatures obtained, with the normal connections and through the selection of the least number of features with the highest detection ability. A fitness function is used to measure the goodness of a attack chromosome. Chromosomes with higher fitness values have a high probability of being selected for crossover to generate offspring attack chromosomes through which newer attacks can be detected. A mutation in-

troduces random changes in the chromosome to explore the possibility of variation of known attacks

7.2 Genetic Algorithm and Intrusion Detection

Genetic algorithm is a powerful domain independent search technique based on principles of evolution and natural selection. Genetic algorithm is generally applied to the optimization problems where the problem is solved using the a application dependent fitness function.

For network intrusion detection with genetic algorithm the "Michigan approach"is selected. In this approach a single chromosome represent a signature or a rule to detect a single attack.A number of these rules will be required for the detection of all the attacks in the data-set. The fitness function evaluates how well a chromosome detects attack and treats normal connections. As the objective clearly states that detection rate of attacks should be maximized where as detection of normal connection as attacks should be minimized. The fitness function being used is taken from [10] and is stated in equation 7.1

$$\frac{\alpha}{A} - \frac{\beta}{B} \tag{7.1}$$

where α (alpha) is the number of connections correctly classified as attacks, β (beta) is the number of normal connections classified incorrectly as attack, **A** is the number

of attacks of that type and **B** is the number of normal connections.

7.3 Simulation Setup

The aim of the simulation was to reverse engineer malicious connections (attacks) in the training set to extract the features and create a chromosome for each attack which is able to detect the same attack in the testing set with the objective of maximizing the true positives and minimizing the false positives.

All the attacks were separately reverse engineered in comparison with the normal connections to generate the attack masks. Attack masks were used to create the attack chromosomes. Only a subset of attack chromosome was provided as input to the search block, this subset was used as initial population by the genetic algorithm was and was evolved using the genetic operators; crossover and mutation.

7.4 Results and Analysis

The results are documented separately for four classes of attacks in the NSL-KDD dataset. Each attack was reverse engineered by comparing the anomalous connection and a normal connection of the similar kind to analyze the difference between them. The parameters which differentiated the two connections were selected for attack detection. The aim was to select the minimum number of features.

For each attack a number of simulations were performed. In each simulation a

different initial population was provided as input and the result was observed over 200 generations on average.

7.4.1 Denial of Service Attacks

7.4.1.1 Smurf

Different samples of the smurf attack and similar normal connections are shown in table 7.1. By reverse engineering, it was determined that the smurf attack specifically occurs with connection parameters of **icmp** as a protocol, **ecr_i** as a service and **SF** as a flag and only with source bytes of **520** and **1032**. So source bytes becomes the malicious differentiator as they have different values in the normal connections.

Connection	Features						
	1	2	3	4	5	6	7 to 41
Smurf	0	icmp	ecr_i	SF	1032	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 80 80 0 0 0 0 1 0 0 255 255 1 0 1 0 0 0 0 0 smurf
Smurf	0	icmp	ecr_i	SF	520	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 252 252 0 0 0 0 1 0 0 255 255 1 0 1 0 0 0 0 0 smurf
Normal	0	icmp	ecr_i	SF	1480	0	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 4 0 0 0 0 1 0 0.5 2 4 1 0 1 0.5 0 0 0 0 normal
Normal	0	icmp	ecr_i	SF	40008	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 0 0 0 0 1 0 0 138 4 0.03 0.06 0.03 0 0 0 0 0 normal
Chromosome	X	icmp	ecr_i	SF	Y	0	X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X smurf

Table 7.1: Chromosome construction for Smurf

The detection behavior of smurf attack with genetic algorithm and reverse engineered features is shown in figure 7.1. The two chromosomes have different values of

Y as in the table 7.1 were provided initially to the system and have the detection of 378 smurf connections. But with the genetic algorithm the final detection is of 665 smurf connections in the test set. It is because the test set contains some different variations of the smurf attack which were recognized using the mutation operator.

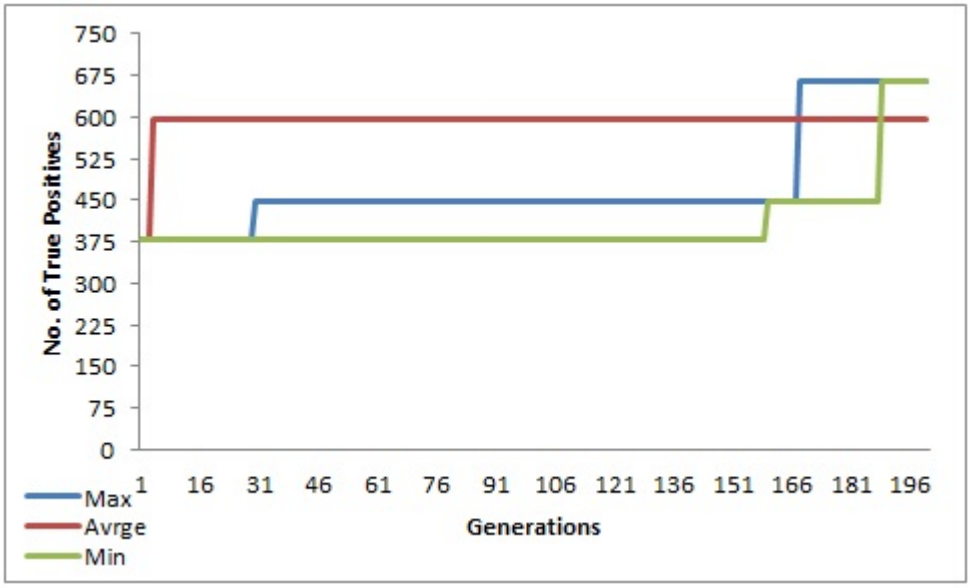

Figure 7.1: Detection of Smurf attack with genetic algorithm

7.4.1.2 Teardrop

A number of samples of the teardrop attack and similar normal connections are shown in table 7.2. By reverse engineering it was determined that the teardrop attack specifically occurs with connection parameters of **udp** as a protocol, **private** as a service and **SF** as a flag and only with source bytes of **28**. So source bytes becomes the malicious differentiator field as they have different values in the normal

connections.

Connection	Features					
	1	2	3	4	5	6	7 to 41
Teardrop	0	udp	private	SF	28	0	0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 10 0 0 0 0 1 0 0 35 10 0.29 0.11 0.29 0 0 0 0 0 teardrop
Teardrop	0	udp	private	SF	28	0	0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 44 44 0 0 0 0 1 0 0 64 44 0.69 0.05 0.69 0 0 0 0 0 teardrop
Normal	0	udp	private	SF	46	48	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 45 45 0 0 0 0 1 0 0 255 254 1 0.01 0.17 0 0 0 0 0 normal
Normal	0	udp	private	SF	53	55	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 509 509 0 0 0 0 1 0 0 255 255 1 0 1 0 0 0 0 0 normal
Chromosome	X	udp	private	SF	28	0	X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X teardrop

Table 7.2: Chromosome construction for Teardrop

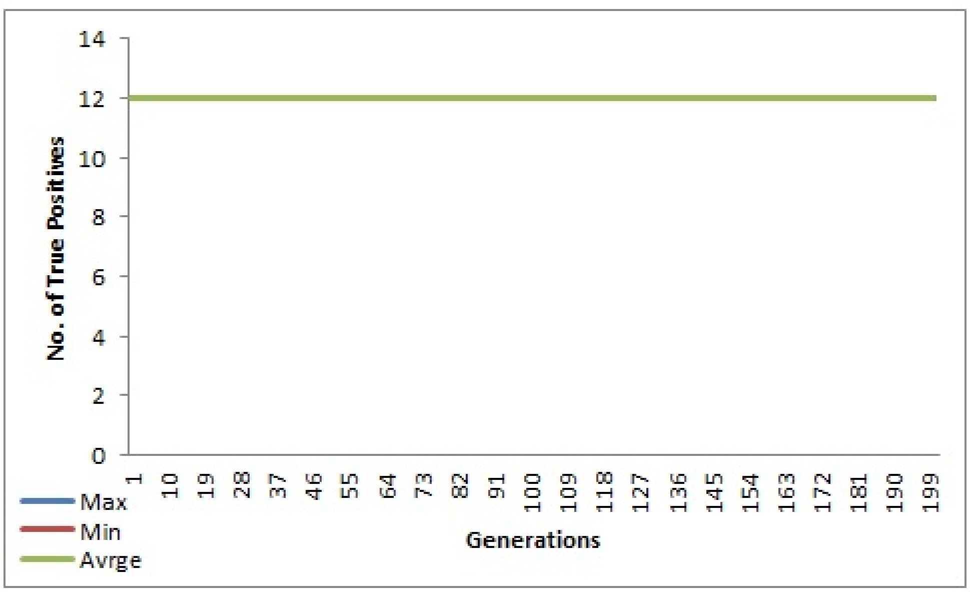

Figure 7.2: Detection of Teardrop attack with genetic algorithm

The detection behavior of teardrop attack with genetic algorithm and reverse engineered features is shown in figure 7.2. The number of detections (true positives) for teardrop remain the same throughout the 200 generations as the instances are exactly the same as the chromosomes provided as initial population and there are also no variations of this attack in the test set. Different runs of the algorithm with different initial population resulted in the same conclusion. With chromosome constructed 37 false positives were also detected.

7.4.1.3 Land

A number of samples of the land attack and similar normal connections are shown in table 7.3. The most prominent feature of the land attack is that it has the same source and destination IP address, which is indicated in the KDD data set by the land field. But to ensure that the number of false positives are minimized, by reverse engineering it was observed for the fields like **tcp** for protocol, **telnet or finger** for service and **S0** for flag, source and destination bytes **0** and the land field of **1**. So the land field becomes the malicious differentiator field as all the normal connections have 0 for this field.

Connection	Features							
	1	2	3	4	5	6	7	8 to 41
Land	0	tcp	finger	S0	0	0	1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 1 1 0 0 1 0 1 255 1 0 0.02 0 0 0 1 0 0 land
Land	0	tcp	telnet	S0	0	0	1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 1 1 0 0 1 0 1 1 3 1 0 1 1 1 1 0 0 land
Normal	0	tcp	finger	SF	8	220	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 147 9 0.04 0.04 0.01 0.22 0 0 0 0 normal
Normal	214	tcp	telnet	SF	167	4647	0	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 255 1 0 0.01 0 0 0 0 0.01 0 normal
Chromosome	X	tcp	finger telnet	SF	0	0	1	X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X land

Table 7.3: Chromosome construction for Land

The detection behavior of teardrop attack with genetic algorithm and reverse engineered features is shown in figure 7.3. The number of detections (true positives) for land attack remain the same throughout the 200 generations as the instances are exactly the same as the chromosomes provided in the initial population. No false alarms are generated in the detection of this attack.

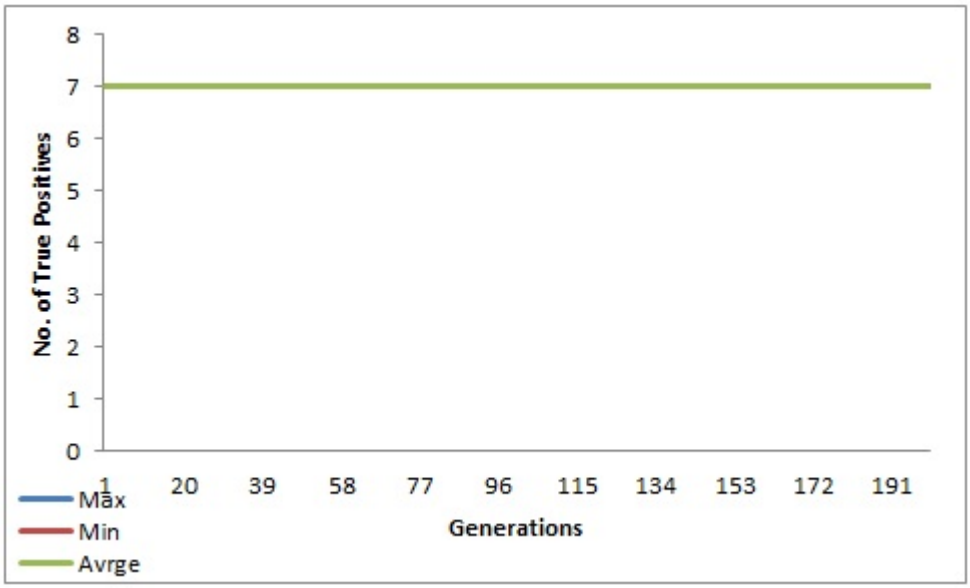

Figure 7.3: Detection of Land attack with genetic algorithm

7.4.1.4 Ping Of Death (POD)

Different samples of the pod attack and similar normal connections are shown in table 7.4. By reverse engineering it was determined that the pod attack can be detected by selecting parameters of **icmp** as a protocol, **ecr_i or tim_i** as a service and **SF** as a flag and source bytes of **1480 and 564** and the destination bytes of **0**. For the pod attack, the source bytes along with protocol and flag are the malicious differentiator fields as they have different values in the normal connections.

Connection	Features						
	1	2	3	4	5	6	7 to 41
Pod	0	icmp	ecr_i	SF	1480	0	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 3 0 0 0 0 1 0 0.67 1 1 1 0 1 0 0 0 0 0 pod
Pod	0	icmp	tim_i	SF	564	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 1 0 0 2 2 1 0 1 0 0 0 0 0 pod
Normal	0	icmp	ecr_i	SF	1480	0	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 4 0 0 0 0 1 0 0.5 2 4 1 0 1 0.5 0 0 0 0 normal
Normal	0	icmp	ecr_i	SF	40008	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 0 0 0 0 1 0 0 138 4 0.03 0.06 0.03 0 0 0 0 0 normal
Chromosome	X	icmp	ecr_i tim_i	SF	Y	0	X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X pod

Table 7.4: Chromosome construction for Pod

The genetic algorithm shows no enhancement in the detection of pod attacks and also in minimizing the number of false positive as a total of 16 normal connections were detected as pod connections where as the other technique have a lower detection rate. The behavior is shown if figure 7.4.

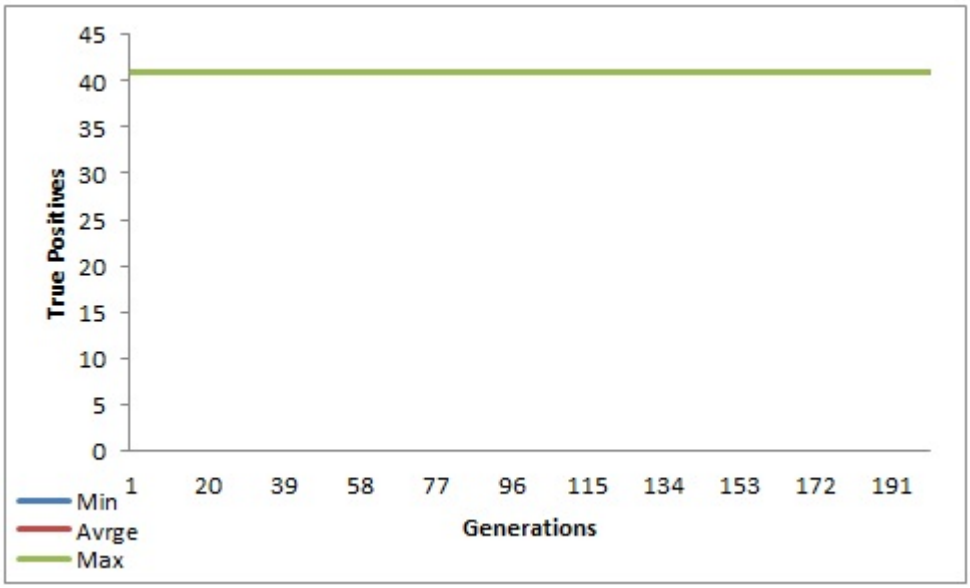

Figure 7.4: Detection of Ping of Death attack with genetic algorithm

7.4.1.5 Back

The connection describing features for the back attack include tcp for **protocol**, **http** for service, **SF or RSTR** for flag and the source and destination bytes are the malicious differentiator with the specific values of 54540, 8314 and 33580, 2920 respectively. The Y in the chromosome in table 7.5 indicates that this field can take several values which are shown in the chromosomes for the detection of this attack.

Connection	Features						
	1	2	3	4	5	6	7 to 41
Back	0	tcp	http	SF	54540	8314	0 0 0 2 0 1 1 0 0 0 0 0 0 0 0 0 4 18 0 0 0 0 1 0 0.11 255 255 1 0 0 0 0 0 0.07 0.07 back
Back	0	tcp	http	RSTR	39240	7300	0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0.5 0.5 1 0 0 255 239 0.94 0.01 0 0 0 0 0.03 0.03 back
Back	0	tcp	http	RSTR	33580	2920	0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0.5 0.5 1 0 0 255 254 1 0.01 0 0 0 0 0.07 0.07 back
Normal	0	tcp	http	SF	209	3376	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 12 0 0 0 0 1 0 0.25 255 255 1 0 0 0 0.01 0.01 0 0 normal
Normal	0	tcp	http	SF	353	1986	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 7 7 0 0 0 0 1 0 0 255 255 1 0 0 0 0 0 0 0 normal
Chromosome	X	tcp	http	SF RSTR	Y	Y	X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X back

Table 7.5: Chromosome construction for Back

Figure 7.5 shows the improvement in the number of detections for the back attack with genetic algorithm. The mutation helps in developing of new chromosomes which lead to further detection of this attack as there are a number of variations for this attack.

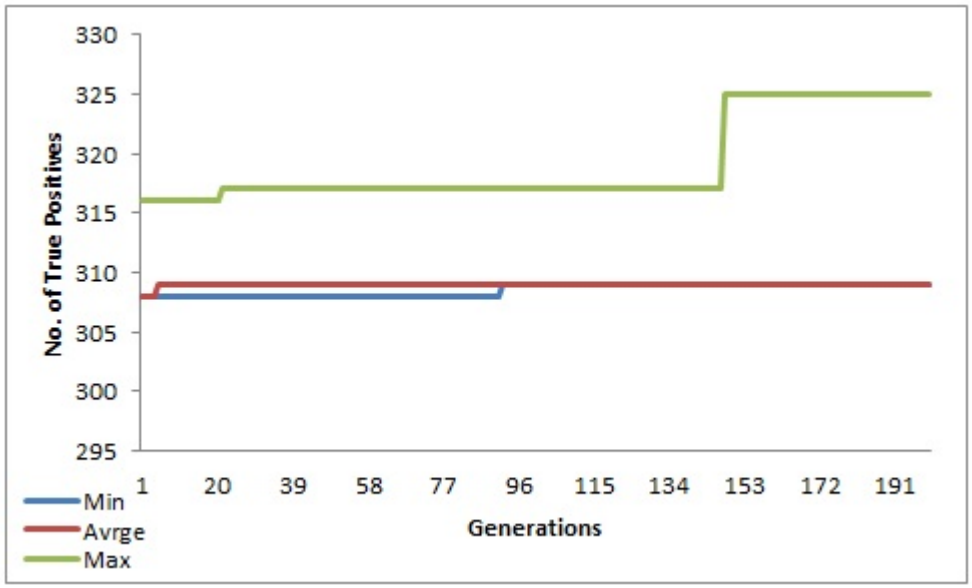

Figure 7.5: Detection of Back attack with genetic algorithm

7.4.1.6 apache2

The apache2 attack is also a form of denial of service attack which was not found in the training set but has a good number of instances in the test set. This attack was detected as a result of cross over between the back attack and neptune attack in one of the simulations. The detection was later confirmed by analyzing the results and the resulting offsprings. The chromosome contrustion example for the detection of apache2 attack is given in table 7.6

A total of 245 apache2 connections were positively detected by the intrusion detection system and the new mutation operator further detected 482 connections of apache2. So from a total of 727 out of 737 apache2 attacks were detected in this

Connection	Features	
	1 to 3	4 to 41
Back	X tcp http	RSTR 54540 8314 0 0 X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
Neptune	0 tcp private	S0 0 0 X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
OffSpring1 (Apache2)	X tcp http	S0 0 0 X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
OffSpring2	0 tcp private	RSTR 54540 8314 0 0 X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X

Table 7.6: Apache2 chromosome construction

way as a number of variations were found with the service field for this attack.

7.4.1.7 Neptune

The neptune attack has a lot different variations in the connection describing features. But through reverse engineering it was analyzed that the source and the destination bytes have a very low value for neptune connections. They are either 0 or 1. So the chromosome was constructed for neptune attack along with its samples and similar normal connections are shown in table 7.7

Connection	Features						
	1	2	3	4	5	6	7 to 41
Neptune	0	tcp	private	REJ	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 256 19 0 0 1 1 0.07 0.06 0 68 2 0.03 0.1 0.01 0 0 0 1 1 neptune
Neptune	0	tcp	Imap4	REJ	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 239 14 0 0 1 1 0.06 0.07 0 255 14 0.05 0.07 0 0 0 0 1 1 neptune
Neptune	0	tcp	telnet	RSTO	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 259 11 0 0 1 1 0.04 0.07 0 255 11 0.04 0.07 0 0 0 0 1 1 neptune
Neptune	0	tcp	time	S0	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 145 1 1 1 0 0 0.01 0.1 0 255 1 0 0.09 0 0 1 1 0 0 neptune
Neptune	0	tcp	netbios_ns	S0	1	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 120 12 1 1 0 0 0.1 0.06 0 255 12 0.05 0.05 0 0 1 1 0 0 neptune
Normal	0	tcp	time	SF	0	4	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0 0 normal
Normal	0	tcp	telnet	SF	129	179	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 255 2 0.01 0.02 0 0 0 0 0 0 normal
Normal	0	tcp	http	SF	306	2239	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 6 6 0 0 0 0 1 0 0 255 233 0.91 0.02 0 0 0 0 0.02 0 normal
Chromosome	X	tcp	Y	Y	1/0	0	X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X neptune

Table 7.7: Chromosome construction for Neptune

The detection behavior of a neptune attack is depicted in figure 7.6. The neptune attack has many variations and for the detection of this attack a very limited number of chromosomes are provided as the initial population and with the help of mutation and crossover the detection percentage reaches upto 99% which is illustrated in the results. This improvement in the detection rate is due to the detection of variation

of the neptune attack as there are many different instances with different values for the service field.

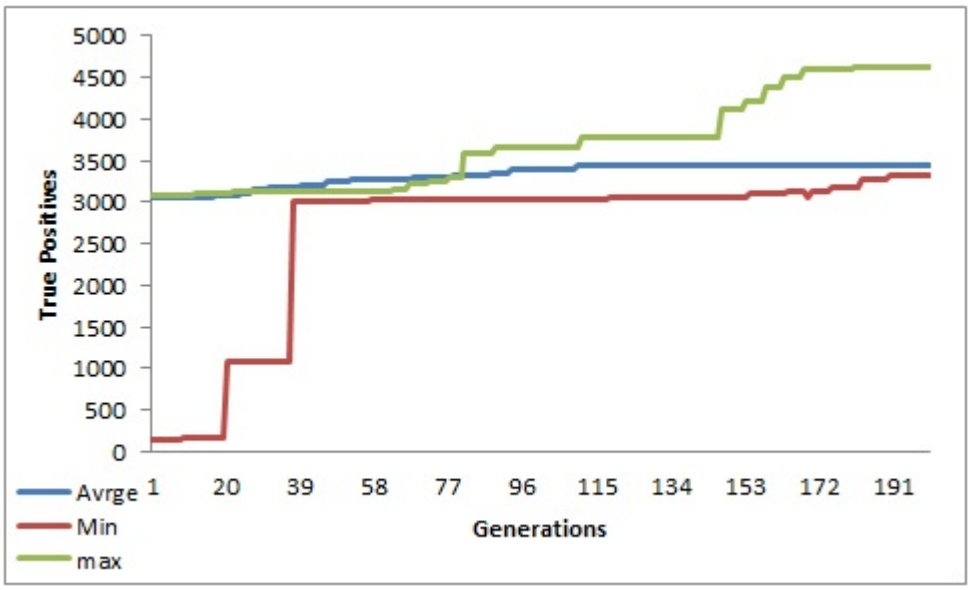

Figure 7.6: Detection of Neptune attack with genetic algorithm

7.4.2 Probe

The probe class of attack have shown a good detection percentage with the pattern based chromosomes because these attacks have constant patterns. Threshold based chromosomes also result in similar detection rate. The satan attack show the best performance in the probe class as with its chromosomes a number os attacks are also detected which were not provided in the initial population. The only problem in the detection of this class of attack is their similarity with the normal connections that is why there is always some number of false positives detected with the malicious

connections.

7.4.2.1 Ipsweep

A number of sample connections of IPsweep attack and similar normal connections are shown is table 7.8. By reverse engineering the features of the attack in comparison with the normal connections the chromosome for the IPsweep is depicted in the table 7.8

From the training set, there were 45 different chromosomes generated for the detection of the ipsweep attack. But when the genetic algorithm was used on all these chromosomes, only two chromosomes were found with positive fitness at the end. The rest of the chromosome had **0** fitness value, which indicated that only two samples of ipsweep were found in the test set. The two successful chromosomes were

1. X icmp eco_i SF 18 0 X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X ipsweep

2. X icmp eco_i SF 8 0 X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X ipsweep

The results are shown in figure 7.7

Connection	Features						
	1	2	3	4	5	6	7 to 41
IPsweep	0	icmp	eco_i	SF	18	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 1 16 1 0 1 1 0 0 0 0 ipsweep
IPsweep	0	icmp	eco_i	SF	8	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 44 0 0 0 0 1 0 1 1 95 1 0 1 0.51 0 0 0 0 ipsweep
IPsweep	0	tcp	private	REJ	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 27 2 0.04 1 0.04 1 0 0 0.89 1 ipsweep
IPsweep	0	tcp	ssh	RSTO	0	15	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 3 2 0.33 1 0.33 1 0 0 1 0.5 ipsweep
Normal	0	tcp	private	REJ	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 56 3 0.02 0.33 0.98 0.67 0.05 0.82 0 65 3 0.05 0.74 0.86 0 0.02 0.33 0.85 0.67 normal
Normal	0	icmp	eco_i	SF	30	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 1 0 0 0 0 0.5 1 0 255 1 0 0.01 0 0 0 0 0 0 normal
Normal	0	icmp	eco_i	SF	10008	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 1 0 0 255 28 0.11 0.02 0.19 0 0 0 0 0 normal
Chromosome	X	icmp	eco_i	SF	18/8	0	X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X ipsweep
Chromosome	X	tcp	Y	SF	0	0	X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X ipsweep

Table 7.8: Chromosome construction for IPsweep

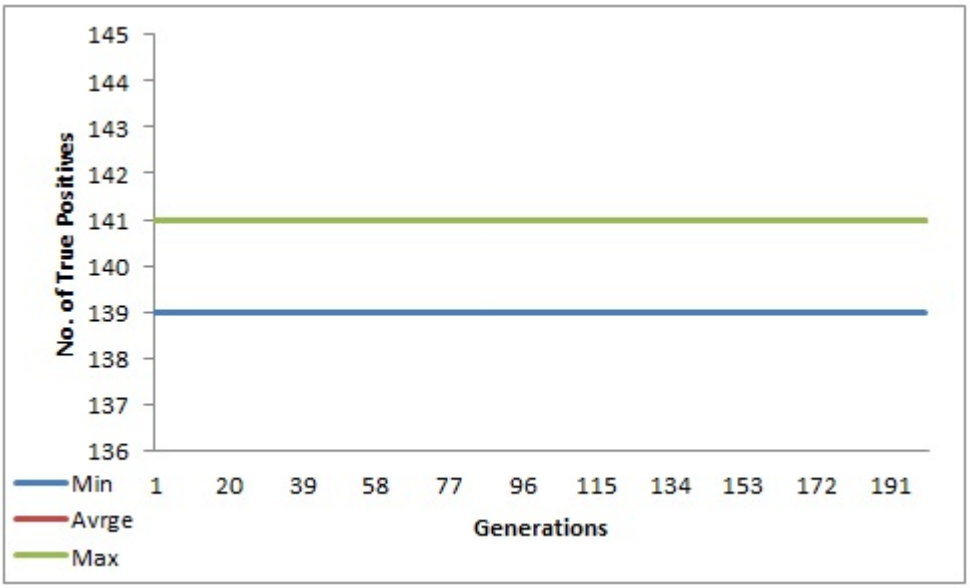

Figure 7.7: Detection of Ipsweep attack with genetic algorithm

7.4.2.2 Portsweep

A portsweep attack is quite similar to ipsweep attack. The reverse engineered chromosomes along with several samples of the portsweep attack and similar normal connections are shown is table 7.9. It can be observed that the selected feature set is similar to ipsweep.

The detection of portsweep attack with genetic algorithm is shown in figure 7.8. As there are different variations of this attack it can be seen that with the runs of genetic algorithm the number of detections is increased and we end up with complete detection of this attack. The new chromosomes generated through both crossover and mutation detect new instances of this attack. Another point to be noted here is the similarity of the normal connection and the attack in the features selected.

Connection	Features					
	1	2	3	4	5	6	7 to 41
Portsweep	0	tcp	private	REJ	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 1 0 0 1 1 0.5 1 0 255 1 0 0.48 0.46 0 0 0 0.47 1 portsweep
Portsweep	0	tcp	other	RSTR	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 255 1 0 0.72 1 0 0 0 1 1 portsweep
Portsweep	0	tcp	ctf	RSTR	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 182 1 0.01 0.46 0.46 0 0 0 0.46 1 portsweep
Normal	0	tcp	other	RSTR	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 255 1 0 0.58 0.56 0 0 0 0.56 1 normal
Normal	0	tcp	private	REJ	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 45 2 0 0 1 1 0.04 0.91 0 54 2 0.04 0.8 0.83 0 0 0 0.83 1 normal
Normal	3	tcp	private	RSTR	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 143 2 0 0 1 1 0.01 0.55 0 150 2 0.01 0.52 0.95 0 0 0 0.95 1 normal
Chromosome	X	tcp	Y	Y	0	0	X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X portsweep

Table 7.9: Chromosome construction for Portsweep

Several different combinations were tried to create a difference but with no success. So it was decided to keep the least number of features which maximized the true positive detection. The number of false positive also increased with with the number of generations which is shown in figure 7.9. The number of false positives generated is still very low.

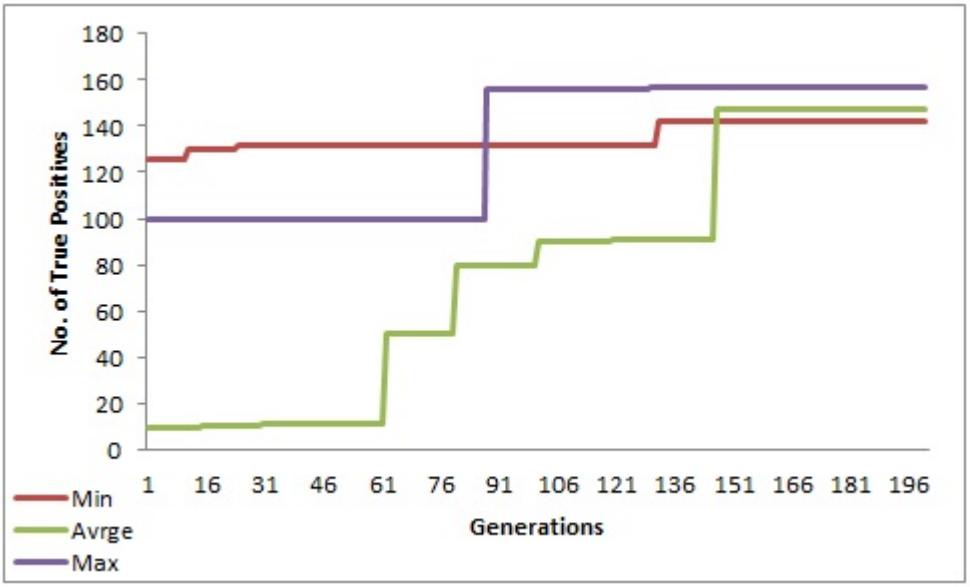

Figure 7.8: Detection of Portsweep attack with genetic algorithm

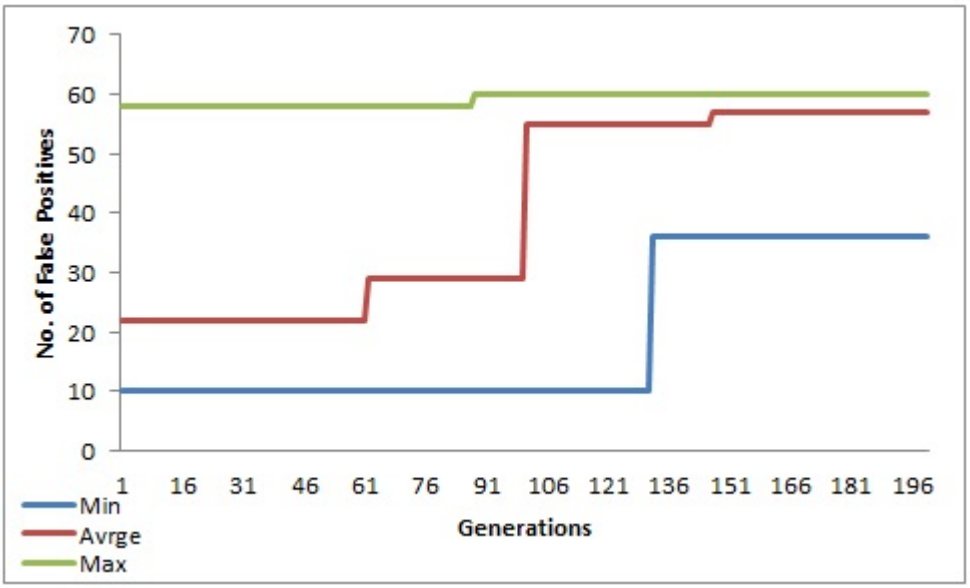

Figure 7.9: FP for Portsweep attack with genetic algorithm

7.4.2.3 Nmap

A comparison of nmap attacks and similar normal connections is shown in table 7.10. It can be seen that for nmap attacks, the first six fields are quite different from the normal connections and they differentiate the anomalous connections from normal connection very well. The SH value in the flag field is only found in the nmap connection, and hence it is the only parameter which can be said to present the anomaly differentiator instead of the whole flag field. The detection behavior of the nmap is shown in figure 7.10. The chromosome detecting most of the attacks was the following

X tcp private SH 0 0 X X X X X X X X X X X X X X X X X X X X X X X X

X X X X X X X X X X X nmap

In one of the simulations, it was not initially provided and was generated as a result of a crossover between an nmap connection and a normal connection. The example is illustrated below where — indicates the cut point.

X tcp finger—SH 0 0 X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X nmap
X tcp private—SF 21 0 X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X normal
X tcp private SH 0 0 X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X nmap
X tcp finger SF 21 0 X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X Anomaly

Connection	Features						
	1	2	3	4	5	6	7 to 41
Nmap	0	icmp	eco_i	SF	8	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 4 0 0 0 0 1 0 1 2 202 1 0 1 0.25 0 0 0 0 nmap
Nmap	0	tcp	private	SH	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 0 249 1 0 0.96 0.97 0 0.97 1 0 0 nmap
Nmap	0	icmp	eco_i	SF	8	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 0 0 0 0 1 0 1 3 39 1 0 1 0.26 0 0 0 0 nmap
Normal	0	icmp	eco_i	SF	30	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 18 18 1 0 1 0 0 0 0 0 normal
Normal	0	udp	private	SF	105	145	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 0 0 0 0 1 0 0 255 254 1 0.01 0.01 0 0 0 0 0 normal
Chromosome	X	tcp	private	SH	0	0	X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X nmap
Chromosome	X	icmp	eco_i	SF	8	0	X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X nmap

Table 7.10: Chromosome construction for Nmap

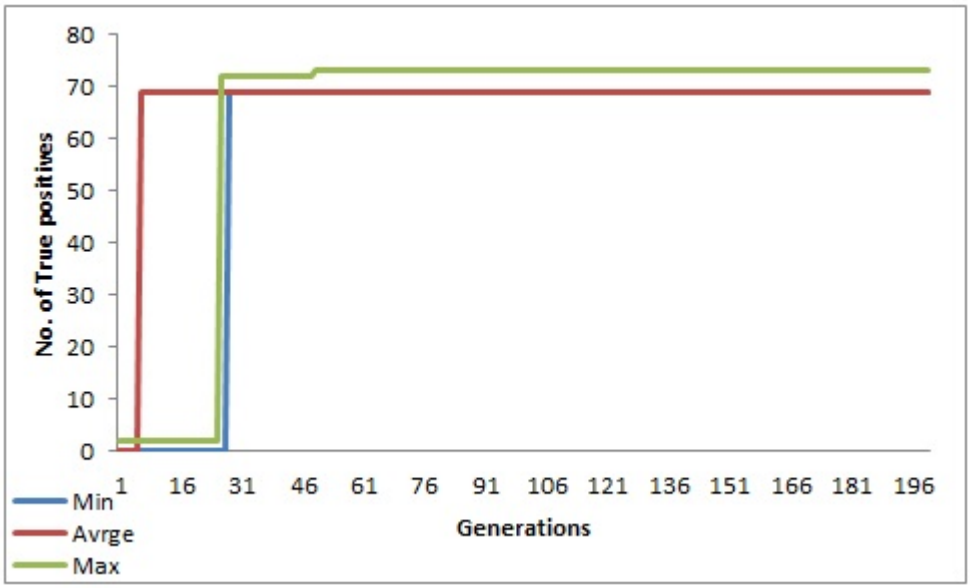

Figure 7.10: Detection of Nmap attack with genetic algorithm

7.4.2.4 Satan

Table 7.11 shows the chromosomes construction steps for satan by comparing it with the normal traffic.

Satan	0	udp	private	SF	1	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 44 8 0 0 0 0 0.18 0.09 0 255 58 0.23 0.21 1 0 0 0 0 0
Satan	0	tcp	private	REJ	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 1 0.08 0 0.9 1 0 1 0 255 1 0 1 0 0 0.09 0 0.86 1
Satan	0	tcp	gopher	SF	6	77	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 1 0 0 0 0 0.5 1 0 255 1 0 0.02 0 0 0 0 0 0
Normal	0	udp	private	SF	105	146	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 229 217 0.95 0.01 0 0 0 0 0 0
Normal	3030	udp	other	SF	147	105	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 1 0 0 255 2 0.01 0.74 0.97 0 0 0 0 0
Chromo some	X	tcp	private	REJ	0	0	X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X satan
Chromo some	X	tcp	other	REJ	0	0	X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X satan
Chromo some	X	tcp/udp	Y	SF/REJ	Y	Y	X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X satan

Table 7.11: Chromosome Construction for Satan

7.4.3 R2L

The problem in the detection of R2L attacks using pattern based chromosomes is the the difference in the connections specifications in the training and the testing sets. This is the reason why the attribute evaluation based methods and subset

evaluation based methods fail to generate a feature subset which results in a good true positive detection. Threshold based methods also fail to improve the detection rate.

7.4.3.1 Ftp_write

Different samples of ftp_write connections and similar normal connections are shown in table 7.12.

Connect.	Features								
	1	2	3	4	5 to 11	12	13	14	15 to 41
Ftp write	32	tcp	ftp	SF	104 449 0 0 0 2 0	1	0	0	0 0 1 0 1 0 0 1 1 1 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0 0
Ftp write	0	tcp	ftp data	SF	613 0 0 0 0 0 0	1	0	0	0 1 0 0 0 0 0 0 0 1 2 0 0 0 0 1 0 1 1 84 1 0 1 0.02 0 0 0 0
Ftp write	67	tcp	login	SF	157 2703 0 0 1 0 0	1	0	0	0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 2 1 0.5 1 0.5 0 0 0 0 0
Normal	0	tcp	ftp data	SF	3720 0 0 0 0 0 0	0	0	0	0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 125 20 0.16 0.03 0.16 0 0 0 0 0

Normal	21	tcp	ftp	SF	241 772 0 0 0 4 0	1	0	0	0 0 0 0 0 0 0 1 2 2 0 0 0 0 1 0 0 255 90 0.35 0.02 0 0 0 0 0.01 0
Chromo some	X	tcp	ftp data	SF	X X X X X X X	1	X	0	X X X X X X X X X X X X X X X X X X X X X X X X X X X ftp_write
Chromo some	X	tcp	login	SF	X X X X X X X	1	X	0	X X X X X X X X X X X X X X X X X X X X X X X X X X X ftp_write

Table 7.12: Chromosome construction for ftp_write

The detection behavior of the ftp_write attack with reverse engineered features and genetic algorithm is shown in figure 7.11. It has been observed that almost all the malicious connections are detected and the number of false positives ranged from a minimum of **0** to a maximum of **365**. The threshold based search yielded a similar result.

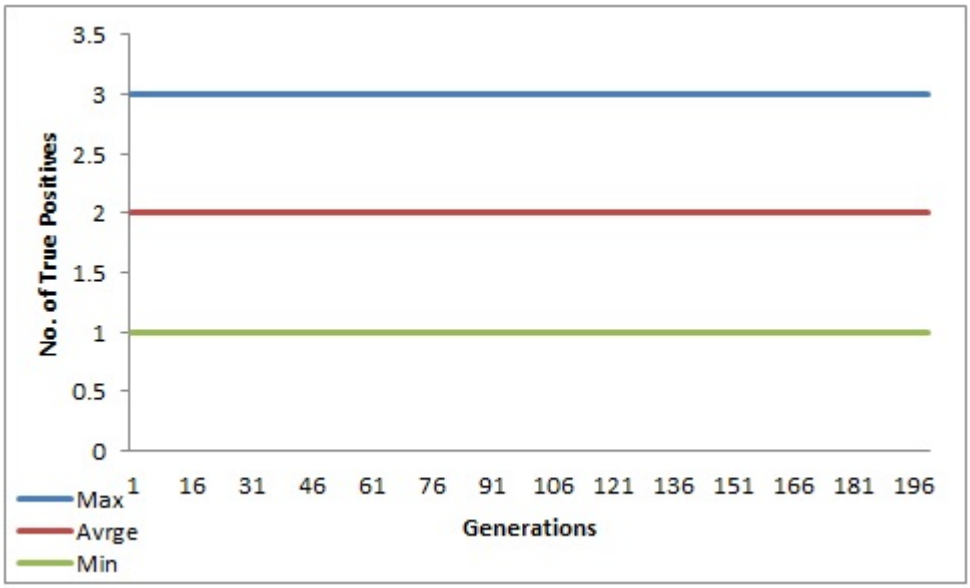

Figure 7.11: Detection of Ftp_write attack with genetic algorithm

7.4.3.2 Warezmaster

A number of samples of warezmaster attack, similar normal network connections and the constructed chromosomes are shown in table 7.13. Warezmaster attack occur with ftp connections both in the control control connections and the data connections. For this attack the **logged_in** field in control ftp connection is **1** whereas in the data connection the same field is equal to **0**.

Connect.	Features							
	1	2	3	4	5	6 to 11	12	13 to 41
Warez master	0	tcp	ftp	SF	36	197 0 0 0 0 0	1	0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 1 0 0 255 1 0 0.05 0 0 0.39 0 0.05 0

Warez master	156	tcp	ftp	SF	950	2551 0 0 0 18 0	1	0 0 0 0 21 0 0 0 0 1 1 1 0 0 0 0 1 0 0 218 1 0 0.03 0 0 0.01 0 0.07 0
Warez master	9	tcp	ftp data	SF	0	5150836 0 0 0 0	0	0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 6 6 1 0 1 0 0 0 0 0
Warez master	10	tcp	ftp data	SF	0	5151385 0 0 0 0	0	0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 5 5 1 0 1 0 0 0 0 0
Warez master	9	tcp	ftp data	SF	0	5151049 0 0 0 0	0	0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 13 13 1 0 1 0 0 0 0 0
Warez master	9	tcp	ftp data	SF	0	5153771 0 0 0 0	0	0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 12 12 1 0 1 0 0 0 0 0
Normal	0	tcp	ftp data	SF	641	0 0 0 0 0	0	0 0 0 0 0 0 0 0 0 0 0 3 2 0 0 0 0 0.67 0.67 0 71 76 0.23 0.07 0.23 0.03 0 0 0 0
Normal	0	tcp	ftp data	SF	3720	0 0 0 0 0	0	0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 125 20 0.16 0.03 0.16 0 0 0 0 0
Normal	21	tcp	ftp	SF	241	772 0 0 0 4	0	1 0 0 0 0 0 0 0 0 0 1 2 2 0 0 0 0 1 0 0 255 90 0.35 0.02 0 0 0 0 0.01 0
Chromo some	X	tcp	ftp	SF	Y	197 X X X X X	1	X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X warezmaster
Chromo some	X	tcp	ftp data	SF	0	X X X X X	0	X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X warezmaster

Table 7.13: Chromosome construction for Warezmaster

The detection of warezmaster with reverse engineered features and genetic algorithm showed a detection of 512 true positives and 35 false positives with the pattern based search. Most of the warezmaster connections with **ftp_data** connection were detected as they they consistently had the source bytes value of 0. On the other hand the connection with the **ftp** service were not detected remarkably.

The threshold based search yielded a similar result for data connections whereas it showed a good detection of ftp connections but with false positives of around 60% as the minimum value was selected and a lot of normal connections were incorrectly classified as warezmaster attacks.

Another important point was observed that for almost all the warezmaster connection based on ftp data connections were detected by threshold search as they had large values as destination bytes without being logged in.

7.4.3.3 Warezclient

No instances of warezclient attack were found in the test set. As this attack is quite similar to warezmaster attack its detection behavior is considered to be the same as the warezmaster attack.

7.4.3.4 Phf

A number of sample connection the phf attack and matching normal connections are shown in table7.14. This is the only attack in R2L category which has proper pattern as it can be seen from the connection table.

No phf connections were found in the test but the generated chromosomes were tested on the traing set itself and it a perfect detection and no false alarms were generated. It means that the chromosomes generated were accurate for the detection of phf attack.

Connect.	Features										
	1	2	3	4	5	6	7-9	10 - 14	15-18	19	20 - 41
phf	0	tcp	http	SF	51	8127	0 0 0	2 0 1 0 1	0 0 0 0	1	0 0 0 1 2 0 0 0 0.5 1 0 1 255 255 1 0 0 0 0 0 0 0
phf	1	tcp	http	SF	51	8127	0 0 0	2 0 1 0 1	0 0 0 0	1	0 0 0 1 1 0 0 0 0 1 0 0 255 246 0.96 0.01 0 0 0 0 0 0
phf	6	tcp	http	SF	51	8127	0 0 0	2 0 1 0 1	0 0 0 0	1	0 0 0 1 1 0 0 0 0 1 0 0 255 249 0.98 0.01 0 0 0 0 0 0
phf	0	tcp	http	SF	51	8127	0 0 0	2 0 1 0 1	0 0 0 0	1	0 0 0 1 1 0 0 0 0 1 0 0 255 241 0.95 0.01 0 0 0 0 0 0
Normal	0	tcp	http	SF	23	8153	0 0 0	0 0 1 0 0	0 0 0 0	0	0 0 0 5 5 0.2 0.2 0 0 1 0 0 30 255 1 0 0.03 0.04 0.03 0.01 0 0.01
Normal	0	tcp	http	SF	20	2790	0 0 0	0 0 1 0 0	0 0 0 0	0	0 0 0 10 27 0 0 0 0 1 0 0.11 48 255 1 0 0.02 0.09 0 0 0 0
Normal	0	tcp	http	SF	306	403	0 0 0	0 0 1 0 0	0 0 0 0	0	0 0 0 38 38 0 0 0 0 1 0 0 38 255 1 0 0.03 0.02 0 0 0 0
Chromo some	X	tcp	http	SF	51	8127	X X X	2 X 1 X 1	X X X X	1	X X X X X X X X X X X X X X X X X X X X X X phf

Table 7.14: Chromosome construction for Phf

7.4.3.5 Guess_Passwd

The guess_password attack proved to be the most difficult attack to detect in the R2L category. The main reason for this is the difference between the instances in the training and the testing set. The example is shown in table 7.15

Dataset	Features
Train	0 tcp telnet RSTO 125 179 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 4 4 1 0 0.25 0 0.25 0.25 0.75 0.75
Train	23 tcp telnet SF 104 276 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 1 2 1 0 1 1 0 0 0 0
Test	0 tcp ftp SF 26 157 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 0 0 233 118 0.51 0.03 0 0 0 0 0 0
Test	1 tcp pop_3 RSTR 15 46 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 255 219 0.86 0.02 0 0 0 0 0.04 0.01
Test	60 tcp telnet S3 120 188 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 0 255 254 1 0.01 0 0 0.02 0.02 0.02 0.02
Test	0 tcp telnet SF 124 174 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 255 197 0.77 0.02 0 0 0 0 0.02 0.03

Table 7.15: Guess_passwd Connection comparison

The pattern based chromosomes were not able to detect a single guess_passwd connection in the test set due to the above mentioned reasons and also a false positive percentage of 0.144 was observed with pattern based chromosomes.
Threshold based chromosomes resulted in a detection rate of over 90% but with the false poitives of about 20% and also a misclassification of 13% was observed.

7.4.4 U2R

The problem with the detection of U2R class of attacks is that such attacks happen when the a normal user tries to gain root access using some commands. But the NSL KDD dataset only contains values and not any command so both the pattern based chromosomes and threshold based chromosomes fail to detect any entry for perl, rootkit and multihop attacks in the test set. U2R attacks which show some detection with the proposed technique are described below:

7.4.4.1 Buffer_overflow

A number of samples of buffer_overflow attacks and similar normal connections are shown in table 7.16. This attack has two connection defining parameters "tcp,ftp_data,SF "and "tcp,telnet,SF "but the 12th and 14th fields of **logged in** and **root shell** really differentiate them from normal connections. But in the dataset there are some of the malicious connections which have the same values like normal connections and are the reason of a number of false positives in the detection of buffer_overflow attack. The true positive detection of buffer_overflow attack is shown in figure 7.12 and the detection of false positives is shown in figure 7.13. The mutation operator is helpful in increasing the true positive detection on the indicated malicious differentiator fields but it also increases the false positives detected. Only 2 different instances of the buffer_overflow attacks are found in the test set.

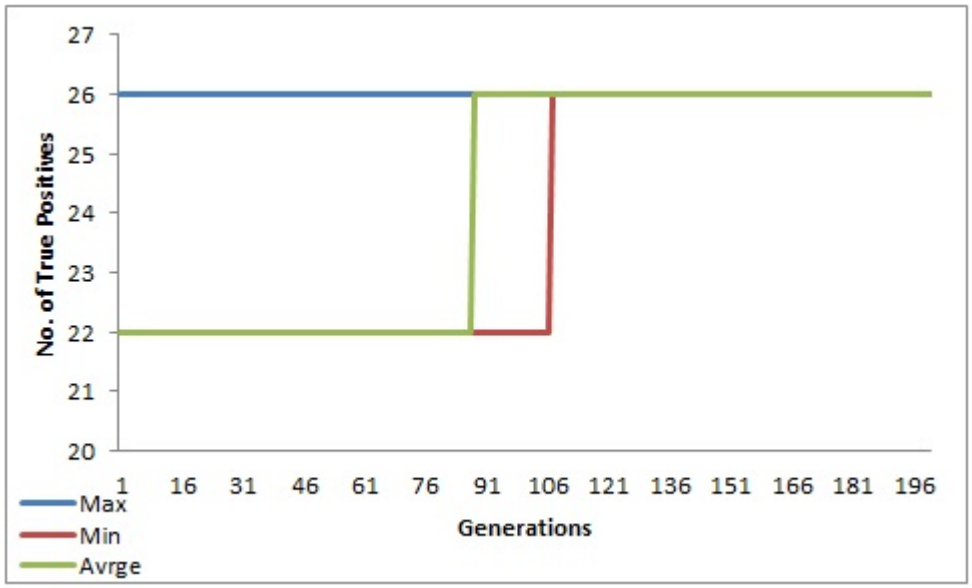

Figure 7.12: Detection of Buffer_overflow attack with genetic algorithm

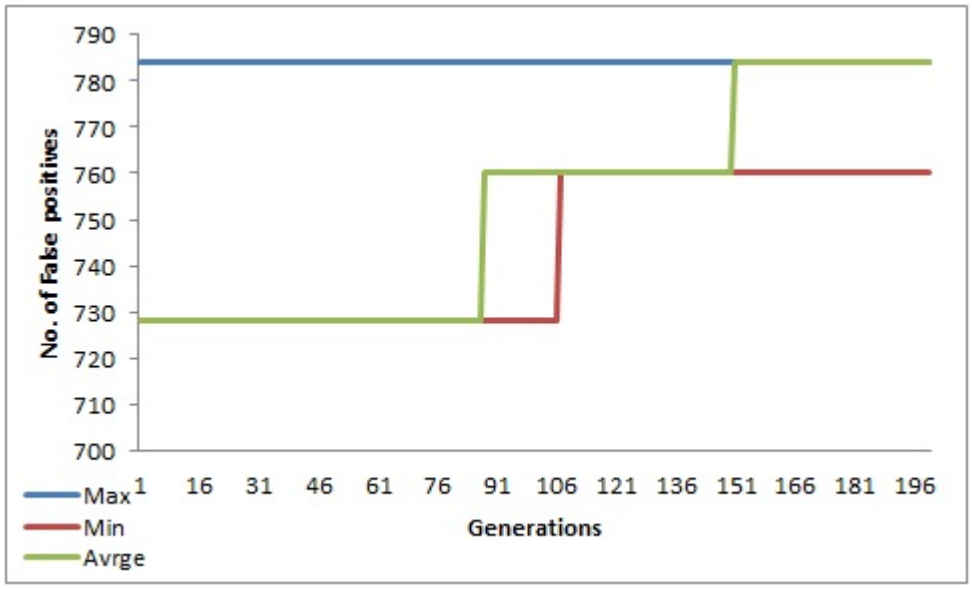

Figure 7.13: FP from Buffer_overflow attack with genetic algorithm

7.4.4.2 Loadmodule

A number of sample loadmodule connections and similar normal connections are shown in table 7.17. It can be seen that loadmodule is described by three connection defining parameter subsets “tcp,ftp,SF”, “tcp,ftp_data,SF”and “tcp,telnet,SF ”. The source bytes value of **0** seems to be the malicious differentiator with tcp,telnet,SF. The su_root value is set for all the Loadmodule connections which the major malicious differentiator of this category.

Cnnction	Features									
	1	2	3	4	5	6 to 11	12	13	14	15 to 41
Load Module	79	tcp	telnet	SF	281	1301 0 0 0 2 0	1	1	1	0 0 4 2 0 0 0 0 1 1 0 0 0 0 1 0 0 1 10 1 0 1 0.3 0 0 0 0.1
Load Module	103	tcp	telnet	SF	302	8876 0 0 0 2 0	1	4	1	0 3 4 2 1 0 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0 0
Load Module	7	tcp	ftp	SF	230	644 0 0 0 4 0	1	0	0	0 0 4 0 0 0 0 0 1 1 0 0 0 0 1 0 0 4 1 0.25 0.75 0.25 0 0 0 0 0
Load Module	0	tcp	ftp data	SF	0	5921 0 0 0 0 0	1	0	0	0 0 0 0 0 0 0 0 2 1 0 0 0 0 0.5 1 0 1 3 1 0 1 0.67 0 0 0 0
Load Module	0	tcp	ftp data	SF	0	2072 0 0 0 1 0	1	0	1	0 0 0 0 0 0 0 0 4 3 0 0 0 0 0.75 0.5 0 3 5 1 0 1 0.4 0 0 0 0

Normal	657	tcp	telnet	SF	648	21257 0 0 0 0 0	1	5	0	0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 255 60 0.24 0.02 0 0 0.79 0.85 0.01 0.03
Normal	140	tcp	telnet	SF	177	27162 0 0 0 0 0	1	6	1	0 3 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 223 1 0 0.02 0 0 0 0 0 0
Normal	22	tcp	ftp	SF	241	771 0 0 0 4 0	1	0	0	0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 0 0 244 68 0.28 0.02 0 0 0 0 0.01 0
Normal	18	tcp	ftp	SF	177	554 0 0 0 3 0	1	0	0	0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 0 0 244 61 0.25 0.03 0 0 0 0 0 0
Normal	0	tcp	ftp data	SF	1766	0 0 0 0 0 0	0	0	0	0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 1 0 0 19 52 0.53 0.11 0.53 0.06 0 0 0 0
Normal	0	tcp	ftp data	SF	641	0 0 0 0 0 0	0	0	0	0 0 0 0 0 0 0 0 3 2 0 0 0 0 0.67 0.67 0 71 76 0.23 0.07 0.23 0.03 0 0 0 0
Chromo some	X	tcp	ftp data	SF	0	X X X X X X	1	X	1	X X X X X X X X X X X X X X X X X X X X X X X X X X X
Chromo some	X	tcp	telnet	SF	281	1301 X X X X X	1	X	1	X X X X X X X X X X X X X X X X X X X X X X X X X X X
Chromo some	X	tcp	ftp	SF	230	644 X X X X X	1	X	0	X X X X X X X X X X X X X X X X X X X X X X X X X X X

Table 7.17: Chromosome construction for Loadmodule

The detection behavior of loadmodule by reverse engineered features showed

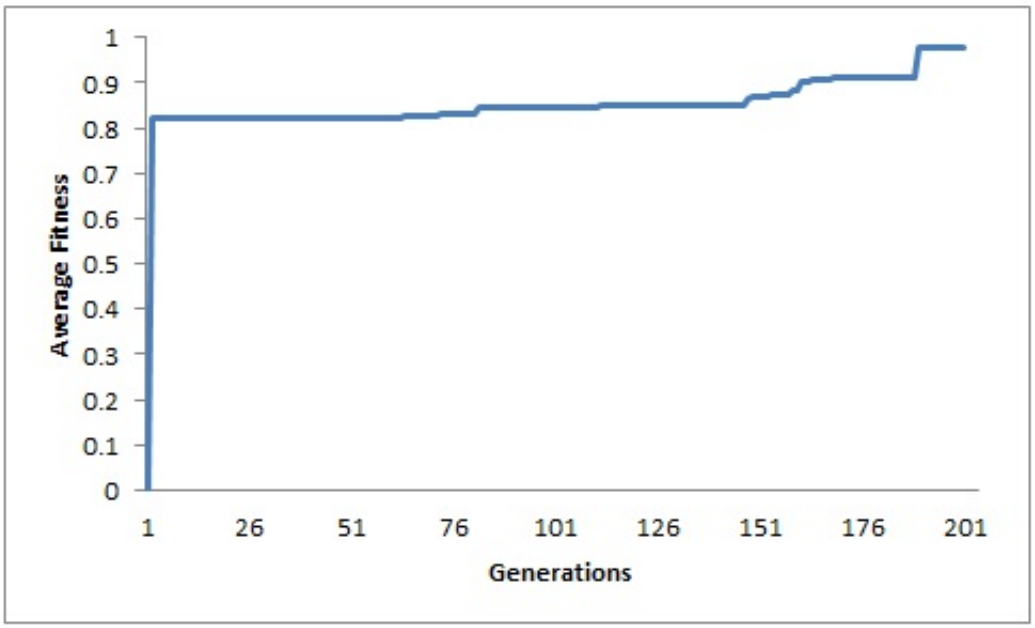

Figure 7.14: Average fitness for DoS attacks

very bad results, As with the pattern based chromosomes not a single loadmodule connection in the test set was detected. On the other hand with threshold based chromosomes 2 correct detection were found along with 2 false negatives and 22 false positives. The genetic algorithm had no effect in increasing the detection rate or reducing the false alarms.

7.4.5 Fitness Analysis

It can be seen from the figure 7.14 and 7.15 that the average fitness of the population increases for denial of service and probe class of attacks as newer attacks are detected. For U2R and R2L class of attack no major improvement of increase in the average fitness is observed.

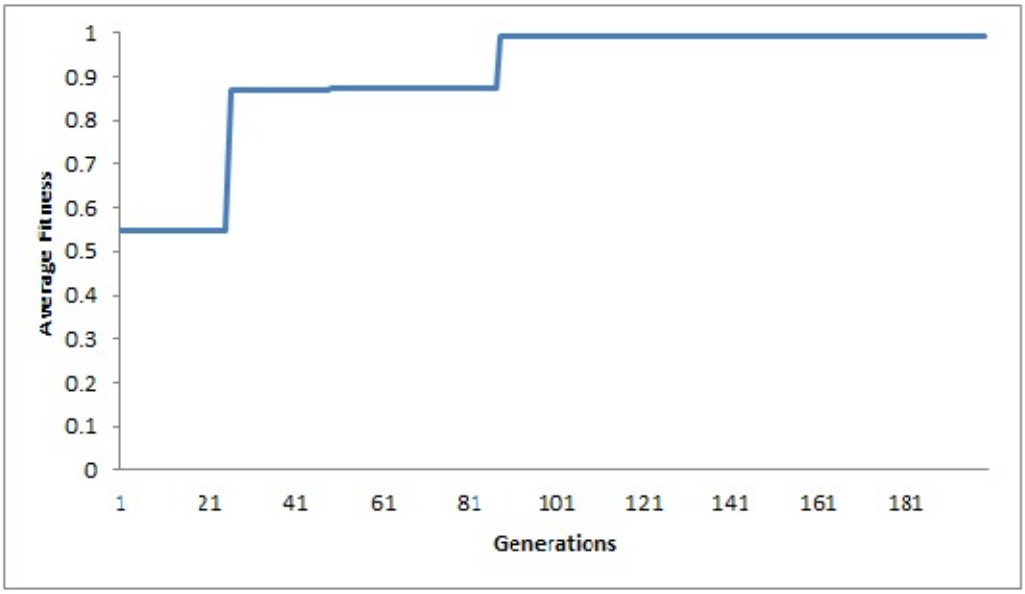

Figure 7.15: Average fitness for Probe attacks

If the number of false negatives are also included in the fitness function then the equation 7.1 takes the form:

$$\frac{\alpha}{A} - \frac{\beta}{B} - \frac{\gamma}{C} \tag{7.2}$$

$$\frac{\alpha}{A} - (\frac{\beta}{B} + \frac{\gamma}{C}) \tag{7.3}$$

where α (alpha) is the number of connections correctly classified as attacks, β (beta) is the number of normal connections classified incorrectly as attack and γ(gamma) is the number of attacks incorrectly classified as different attacks. **A** is the number of attacks of that type and **B** is the number of normal connections and **C** is the total number of false negatives possible. It was observed that by accounting for

false negatives in the fitness does effect the detection rate for DoS attacks as there is no false negative detected. For probe attacks only portsweep showed the detection of some false negatives but the the value was so small that it can be considered negligible. For U2R and R2L attacks a similar trend was observed.

7.5 Comparative Analysis

In this section we analyze the results of implemented feature selection techniques and compare them with our genetic algorithm based network intrusion detection technique. As the simulation is based on different attacks so the results are documented based on the attack category.

7.5.1 Denial of Service

If we analyze the results for denial of service attack, it can be concluded that our proposed GA based technique detects all the smurf attack connections in the test set whereas the other techniques only detected up to 60% of these attacks. For the back attack, the best detection rate is shown by the ranking method but with a higher number of false alarms, but the GA based method exhibits a lower detection rate. For neptune attack, the detection rate is the highest with our proposed method and the false alarm rate is also minimized. The results are similar for land attack and teardrop attack; as the maximum detection rate and minimum false alarm rate are

observed with our proposed GA based technique. The results for the DoS attacks are summarized in table 7.18 and figure 7.16 and figure 7.17 shows the comparison of detection rates for denial of service attacks.

Attack	Ranking Method		Subset Evaluation Method				GA based technique		Improvement
			Simulation 1		Simulation 2				
	Detection	False Alarm	Detection	False Alarm	Detection	False Alarm	Detection	False Alarm	
smurf	56.84	0	56.84	0	60.12	0	100	0	39.87
back	100	2.15	94.42	0	19.49	0	91.92	0	2.15
neptune	97.68	0.32	8168	0	11.89	0	100	0.24	2.31
pod	100	0.38	100	99.61	60.97	0.12	100	0.16	0.21
teardrop	100	0.16	100	0.53	0.16	1.02	100	0.38	0
land	57.14	0	100	0	57.14	0	100	0	42.85

Table 7.18: Detection Rate(%) for DoS Attacks

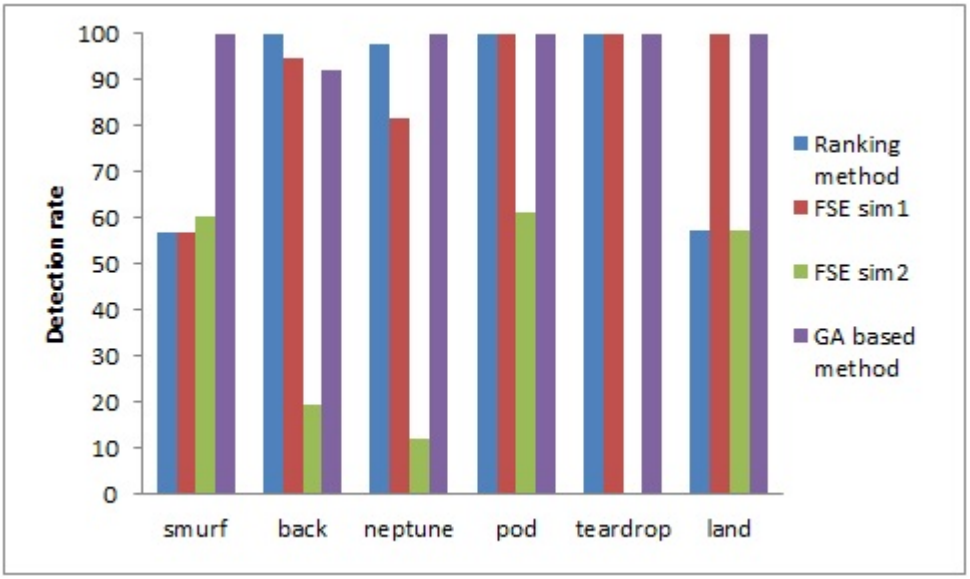

Figure 7.16: Detection rate comparison for DoS attacks

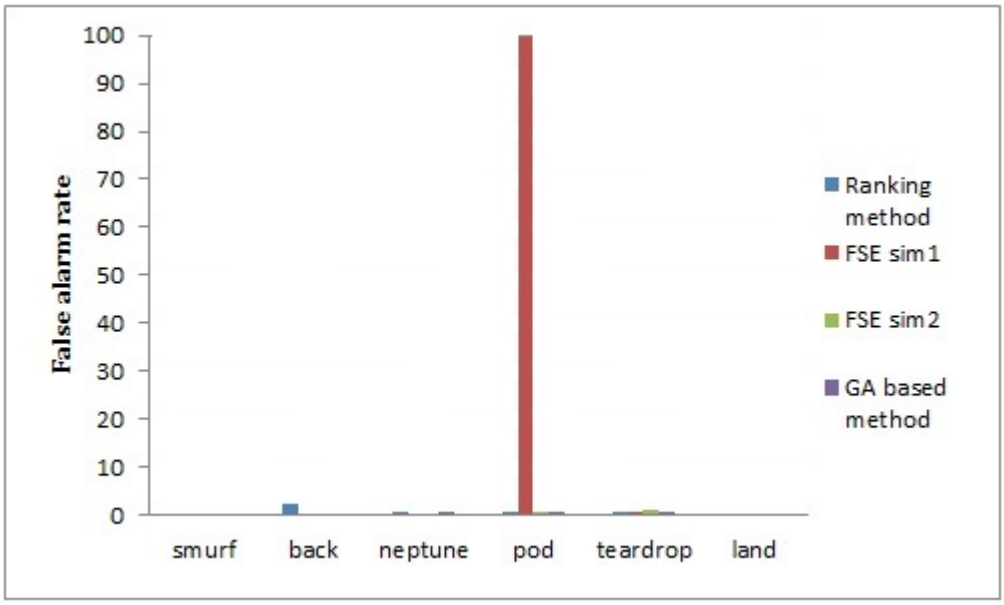

Figure 7.17: False alarm rate comparison for DoS attacks

7.5.2 Probe

For the probe class of attacks, our proposed GA based network intrusion detection technique shows an increased detection rate for satan attack, portsweep attack, and nmap attack. The false alarm rate for these attacks is also the minimum with our proposed technique. The maximum detection rate for ipsweep is observed with all the techniques, but our proposed scheme shows a lower false alarm rate as compared to the filtered subset evaluation, and a higher false alarm rate compared to the ranking method. Overall, our proposed GA method shows a higher detection rate and a lower false alarm rate. The results for probe class of attacks are summarized in table 7.19 and figure 7.18 and figure 7.19 show the comparison plots.

Attack	Ranking Method		Subset Evaluation Method				GA based technique		Improvement
			Simulation 1		Simulation 2				
	Detection	False Alarm	Detection	False Alarm	Detection	False Alarm	Detection	False Alarm	
satan	57.41	0.46	48.43	2.09	1.22	0	83.94	1.64	26.53
portsweep	98.08	5.87	73.24	0.31	15.92	0	99.36	0.61	1.27
ipsweep	100	0.78	100	1.76	98.58	0	100	1.00	0.75
nmap	82.19	0	80.82	0	2.73	0	100	0	17.80

Table 7.19: Detection Rate(%) for Probe Attacks

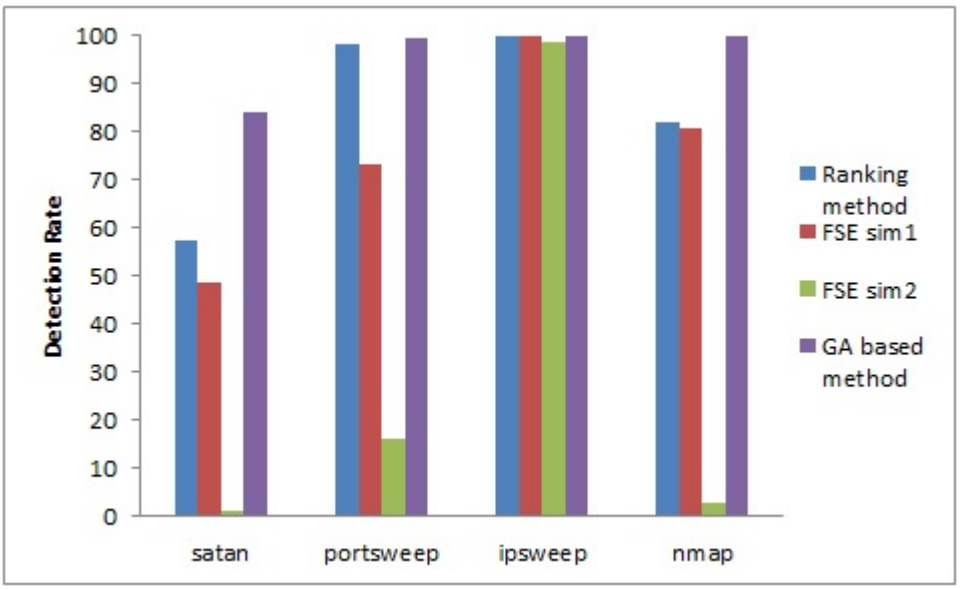

Figure 7.18: Detection rate comparison for Probe attacks

7.5.3 User to Root (U2R)

In the U2R attacks, an attacker tries to exploit the system vulnerabilities by executing commands. The KDD dataset only contain values, and therefore these U2R attacks are hard to detect with only the values from the dataset. The detection

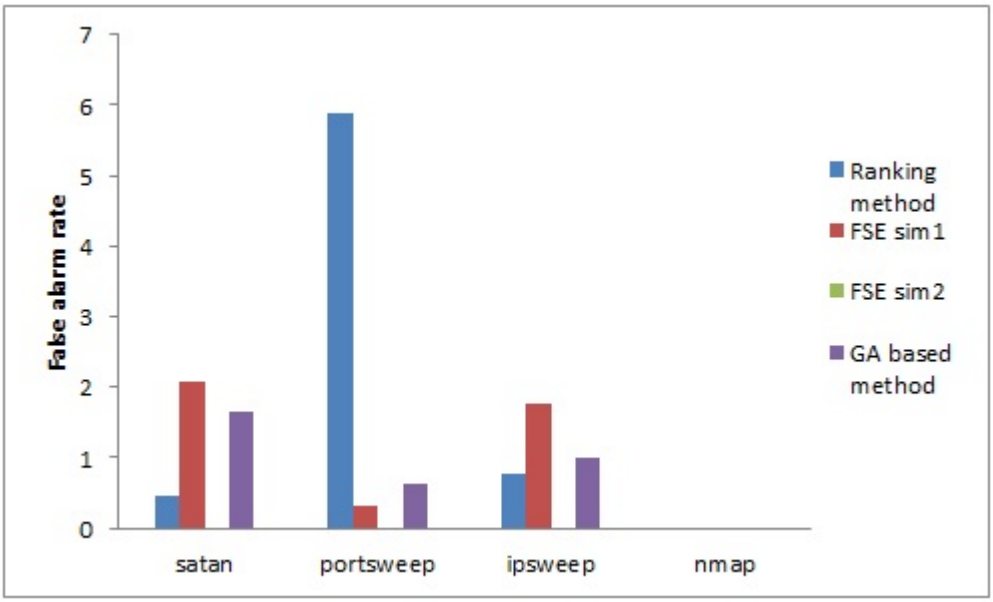

Figure 7.19: False alarm rate comparison for Probe attacks

rates for buffer overflow attack and perl attack are higher as compared to other techniques, and also the false alarms for these attacks are lower with our proposed technique. The rootkit attack was not detected by our proposed technique but it also shows no false alarms, which are higher with all the other techniques. The reason for this behavior was the absence of a symmetric behavior as the instances in the training and the test sets are different for this attack so it is difficult for the system to detect attacks in the test set based of rules developed from the training set. Similar results are shown for the multihop attack as well. The results for the U2R attacks are summarized in table 7.20 and figure 7.20 and figure 7.21 shows the comparison of results.

Attack	Ranking Method		Subset Evaluation Method				GA based technique		Improvement
			Simulation 1		Simulation 2				
	Detection	False Alarm	Detection	False Alarm	Detection	False Alarm	Detection	False Alarm	
rootkit	1.21	99.93	92.30	100	0	0	0	0	0
buffer overflow	60	99.61	100	100	0	0	90	0.89	30
perl	50	0	0	0	0	0	100	0.41	50
loadmodule	100	100	100	100	0	0	34.19	5.78	94.21
multihop	88.88	99.97	83.33	99.92	0	0	12.34	11.12	88.85

Table 7.20: Detection Rate(%) for U2R Attacks

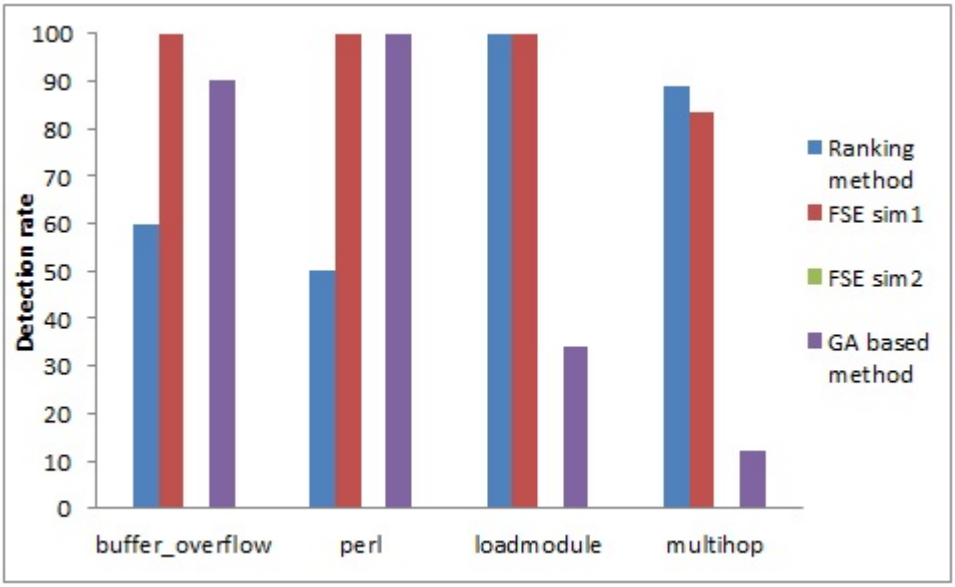

Figure 7.20: Detection rate comparison for U2R attacks

7.5.4 Remote to Local (R2L)

Our proposed GA based method showed a higher detection rate for warezmaster attack with an increased false alarm rate as compared to other techniques. The

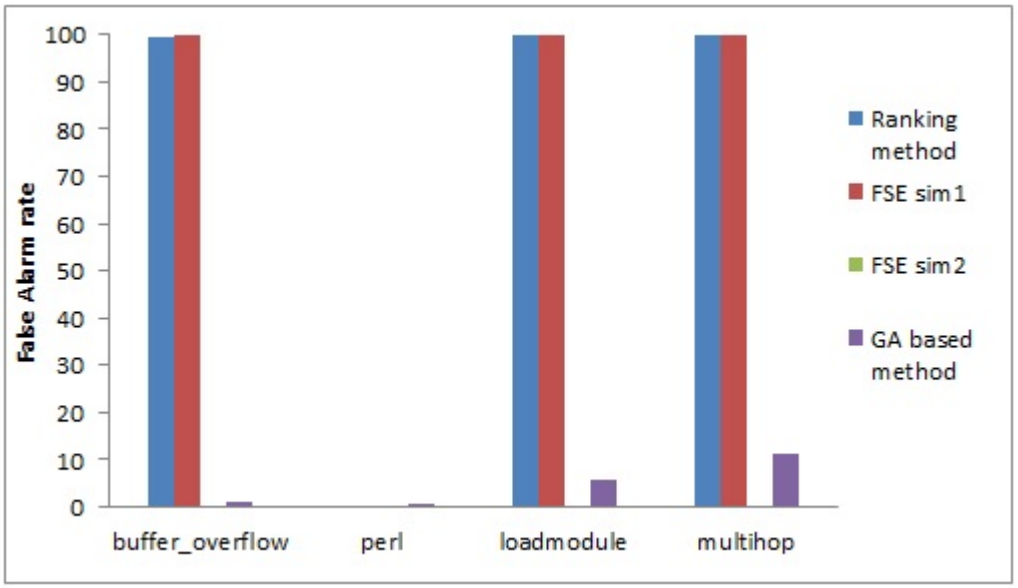

Figure 7.21: False alarm rate comparison for U2R attacks

maximum detection rate of less than 2% was found for the warezmaster attack with ranking method, but our proposed method shows a detection rate of 54%. A good detection rate is also shown for guess passwd attack with a very low false alarm rate with our proposed technique. A smaller number of false positives are observed for ftp write attack. For phf attack a higher detection rate and a lower false positive rate is observed. Table 7.21 show the results for R2L attacks, figure 7.22 and figure 7.23 show the comparison of techniques.

Attack	Ranking Method		Subset Evaluation Method				GA based technique		Improvement
			Simulation 1		Simulation 2				
	Detection	False Alarm	Detection	False Alarm	Detection	False Alarm	Detection	False Alarm	
warezmaster	1.90	0.051	0	0.051	0	0	54.23	5.12	52.33
guess passwd	0	0	100	99.96	0	0	62.14	0.14	62.14
ftp write	0	3.78	100	100	0	0	100	3.75	0.03
phf	50	0.03	50	0.18	0	0	72.14	0	22.14

Table 7.21: Detection Rate(%) for R2L Attacks

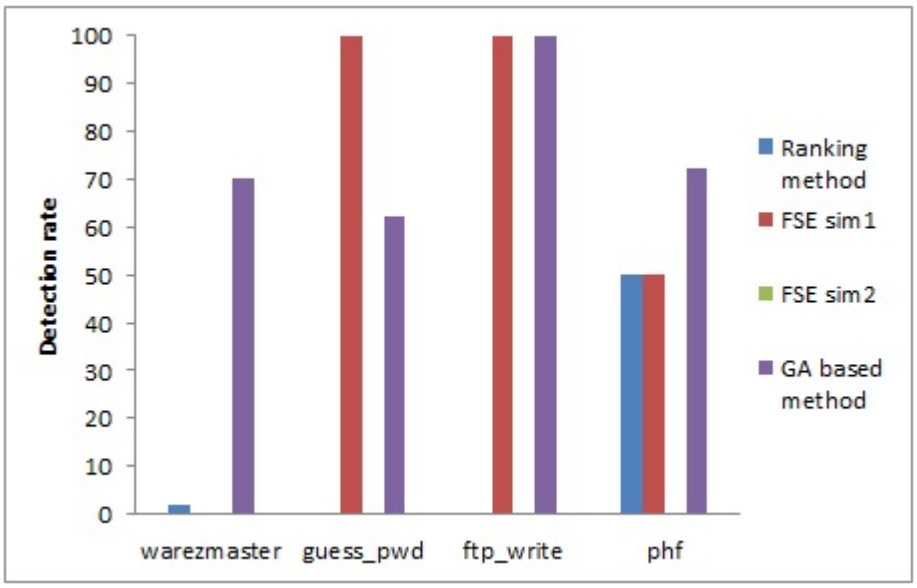

Figure 7.22: Detection rate comparison for R2L attacks

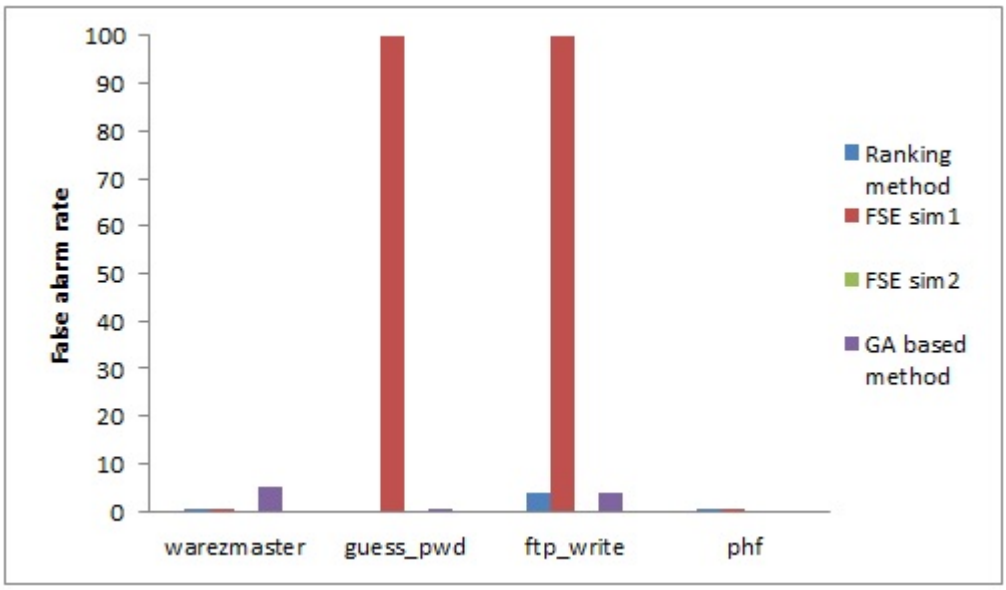

Figure 7.23: False alarm rate comparison for R2L attacks

Cnnction	Features							
	1	2	3	4	5 to 11	12	13	14	15 to 41
Buffer overflow	0	tcp	ftp_data	SF	0 5696 0 0 0 0 0	1	0	0	0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 1 81 1 0 1 0.02 0 0 0 0
Buffer overflow	290	tcp	telnet	SF	415 70529 0 0 0 3 0	1	4	0	0 4 4 0 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0 0
Normal	49	tcp	telnet	SF	2402 3939 0 0 0 40	1	2	1	0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 1 2 1 0 1 1 0 0 0 0
Normal	0	tcp	ftp_data	SF	16196 0 0 0 0 0	0	0	0	0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 31 22 0.26 0.1 0.26 0.14 0 0 0 0
Normal	0	tcp	ftp_data	SF	3136 0 0 0 0 0 0	0	0	0	0 0 0 0 0 0 0 0 2 2 0 0 0 0 1 0 0 198 63 0.32 0.04 0.32 0 0 0 0 0
Normal	39	tcp	telnet	SF	270 8652 0 0 0 0 0	0	0	0	0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 210 25 0.12 0.08 0 0 0 0 0.01 0
Chromo some	X	tcp	telnet	SF	X X X X X X X	1	X	1	X X X X X X X X X X X X X X X X X X X X X X X X X X X buffer_overflow
Chromo some	X	tcp	ftp_data	SF	X X X X X X X	1	X	0	X X X X X X X X X X X X X X X X X X X X X X X X X X X buffer_overflow
Chromo some	X	tcp	telnet	SF	X X X X X X X	1	X	1	X X X X X X X X X X X X X X X X X X X X X X X X X X X buffer_overflow
Chromo some	X	tcp	ftp_data	SF	X X X X X X X	1	X	1	X X X X X X X X X X X X X X X X X X X X X X X X X X X buffer_overflow

Table 7.16: Chromosome Construction for buffer_overflow

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In this thesis work I have presented a network intrusion detection system based on genetic algorithm. The objective of the system is to enhance the detection of attacks while reducing the number of false alarms. Genetic algorithm is an iterative heuristic based on the process of natural selection and evolution. A reverse engineering based feature selection method is used for selecting the appropriate features for generating chromosomes for detection of attacks in the data set. These chromosomes are provided as initial population to the system along with a number of instances of other attacks and normal connections. The system proceeds by evolving these chromosomes using genetic operators crossover and mutation. The detection rates of all these attacks were analyzed on 200 generations of the algorithm. The system

shows excellent detection rate for denial of service and probe classes of attacks and good results for u2r and r2l categories.

The proposed scheme is also capable of detecting variants of known anomalous connections and also some other unknown attacks.

The three parameters of intrusion detection systems are also enhanced in the proposed scheme Speed of the system is directly related to the the number of comparisons it takes on rule to detect a connection for being anomalous. As all the attacks are detected with the least number of features so it can be said the the proposed scheme is quick in detecting attacks.

Accuracy is quite high for denial of service and probes classes and is good for u2r and r2l.

The system is self adaptable for new attacks in some cases just the chromosome is needed for the system to detect new attacks.

8.2 Future Directions

As it was observed that the proposed scheme was able to detect attacks which are not provided as training so it is possible that the scheme can be used for the detection of zero day attacks. But to ensure that this provides good results the data set should be expanded to include all the attacks that are known today.

Intrusion detection systems need to be updated for new attacks so new research

methods and different algorithms can be applied for the solution of this problem.

Bibliography

[1] Nikolai Bezroukov. Intrusion detection general issues, July 2003.

[2] Sadiq M. Sait and Habib Youssef. *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1st edition, 1999.

[3] Sadiq M. Sait and Habib Youssef. *VLSI physical design automation - theory and practice.* IEEE, 1995.

[4] Alex A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[5] Ricardo H. C. Takahashi, Kalyanmoy Deb, Elizabeth F. Wanner, and Salvatore Greco, editors. *Evolutionary Multi-Criterion Optimization - 6th International Conference, EMO 2011, Ouro Preto, Brazil, April 5-8, 2011. Proceedings*, volume 6576 of *Lecture Notes in Computer Science*. Springer, 2011.

[6] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1-2):18–28, 2009.

[7] Nong Ye and Qiang Chen. An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. *Quality and Reliability Engineering International*, 17(2):105–112, March 2001.

[8] J.M. Estevez-Tapiador, P. Garcia-Teodoro, and J.E. Diaz-Verdejo. Stochastic protocol modeling for anomaly based network intrusion detection. In *Information Assurance, 2003. IWIAS 2003. Proceedings. First IEEE International Workshop on*, volume 02798, pages 3–12. IEEE, 2003.

[9] Zorana Banković, Slobodan Bojanić, and Octavio Nieto-taladriz. Evaluating Sequential Combination of Two Genetic Algorithm-Based Solutions for Intrusion Detection. *Machine Learning*, pages 147–154, 2009.

[10] Adhitya Chittur. Model generation for an intrusion detection system using genetic algorithms. *High School Honors Thesis, Ossining High School. In*, 2001.

[11] S.J. Stolfo, P.K. Chan, E. Eskin, M. Miller, and S. Hershkop. Real time data mining-based intrusion detection. *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, 1:89–100, 2001.

[12] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[13] Wei Li. Using genetic algorithm for network intrusion detection. *Proceedings of the United States Department of Energy Cyber Security Group*, 2004.

[14] Zorana Banković, J.M. Moya, Álvaro Araujo, Slobodan Bojanić, and O. Nieto-Taladriz. A Genetic Algorithm-based Solution for Intrusion Detection. *Journal of Information Assurance and Security*, 4:192–199, 2009.

[15] Olusegun Folorunso and Olufunke R Vincent. ID-SOMGA : A Self Organising Migrating Genetic Algorithm-Based Solution for Intrusion Detection. *Science*, 3(4):80–92, 2010.

[16] R. H. Gong, M. Zulkernine, and P. Abolmaesumi. A Software Implementation of a Genetic Algorithm Based Approach to Network Intrusion Detection. *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks (SNPD/SAWN'05)*, pages 246–253, 2005.

[17] A.B.M Alim Al Islam, Md. Ariful Azad, Md. Khurshid Alam, and Md. Shamsul Alam. Security Attack Detection using Genetic Algorithm (GA) in Policy Based

Network. *2007 International Conference on Information and Communication Technology*, pages 341–347, March 2007.

[18] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and A.A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009*, number Cisda, pages 1–6, 2009.

[19] Kristopher Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Bachelors thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 1999.

[20] Land attack description, March 2012.

[21] Land attack description, March 2012.

[22] Description of neptune attack, February 2012.

[23] Description of ping of death attack, February 2012.

[24] Description of smurf attack, February 2012.

[25] Teardrop attack description, February 2012.

[26] Udpstorm attack description, February 2012.

[27] Loadmodule attack description, March 2012.

[28] Gene Spafford Simson Garfinkel. Practical unix and internet security. 1996.

[29] Mscan description, March 2012.

[30] Mscan description, March 2012.

[31] Nmap homepage, March 2012.

[32] Description of saint, March 2012.

[33] Description of satan attack, March 2012.

[34] Description of ftp write attack, March 2012.

[35] Description of phf attack, March 2012.

[36] Description of phf attack, March 2012.

[37] Weka Machine Learning Project. Weka. URL http://www.cs.waikato.ac.nz/~ml/weka.

[38] Shaomin Wu and Peter A Flach. Feature Selection with Labelled and Unlabelled Data.

[39] Shina Sheen. Network Intrusion Detection using Feature Selection and Decision tree classifier. *Time*.

[40] Khaled Labib. Computer security and intrusion detection. *Crossroads*, 11(1):2–2, September 2004.

[41] SDS Siva. Discriminant Analysis based Feature Selection in KDD Intrusion Dataset. *International Journal of Computer Applications*, 31(11):1–7, 2011.

[42] Noris Mohd Norowi and Nasir B Sulaiman. A Study on Feature Selection and Classification Techniques for. *Computer*, pages 331–336, 2008.

[43] FY Leu. A DoS/DDoS Attack Detection System Using Chi-Square Statistic Approach. *iiisci.org*, 2007.

[44] Bo Zhou, Qi Shi, Madjid Merabti, Byrom Street, United Kingdom, Email Bzhouljmuacuk, Q Shi, and M Merabti. Intrusion Detection in Pervasive Networks Based on a Chi-Square Statistic Test . *Computer*, 2006.

[45] Eldad Eilam. *Reversing: Secrets of Reverse Engineering*. Wiley, 2005.

[46] Wenke Lee and SJ Stolfo. Mining audit data to build intrusion detection models. *Proceeding of the Fourth International Conference on Knowledge Discovery and Data Mining*, 1998.

[47] Sadiq M. Sait & Habib Youssef. *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems*. IEEE Computer Society Press Los Alamitos, CA, USA, 1999.

[48] HA, JH Jahnke, and DB Smith. Reverse engineering: a roadmap. *Software Engineering*, 2000.

[49] E. Chikofsky and J. Cross. Reverse engineering and design recovery: A taxonomy. *IEEE Software*, 7(1):13–17, January 1990.

[50] Thomas E. Dube, Bobby D. Birrer, Richard A. Raines, Rusty O. Baldwin, Barry E. Mullins, Robert W. Bennington, and Christopher E. Reuter. Hindering reverse engineering: Thinking outside the box. *IEEE Security and Privacy*, 6:58–65, 2008.

[51] S. Rosset, Claudia Perlich, and Y. Liu. Making the most of your data: KDD Cup 2007 How Many Ratings winner's report. *ACM SIGKDD Explorations Newsletter*, 9(2):66–69, 2007.

[52] Introduction This, Name Type, S H R Description, Intrinsic Attributes, Automatic Classification, and Computer Architecture. GureKddcup database description. 99:1–6.

[53] Ming-Yang Su, Sheng-Cheng Yeh, Chun-Yuen Lin, and Chen-Han Tsai. Using Genetic Algorithm to Improve an Online Response System for Anomaly Traffic by Incremental Mining. *International Symposium on Parallel and Distributed Processing with Applications*, pages 582–587, September 2010.

[54] Fred Cohen. Computer viruses:: Theory and experiments. *Computers & security*, 6(1):22–35, 1987.

[55] Varun Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):1–58, 2009.

[56] Marina Thottan and Chuanyi Ji. Anomaly detection in IP networks. *Signal Processing, IEEE Transactions on*, 51(8):2191–2204, 2003.

[57] Shelly Xiaonan Wu and Wolfgang Banzhaf. The use of computational intelligence in intrusion detection systems: A review. *Applied Soft Computing*, 10(1):1–35, January 2010.

[58] P. Helman, G. Liepins, and W. Richards. Foundations of intrusion detection (computer security). *[1992] Proceedings The Computer Security Foundations Workshop V*, pages 114–120, 2008.

[59] S. Akbar, K.N. Rao, and JA Chandulal. Intrusion detection system methodologies based on data analysis. *International Journal of Computer Applications IJCA*, 5(2):10–20, 2010.

[60] Maheshkumar Sabhnani and G. Serpen. KDD feature set complaint heuristic rules for R2L attack detection. In *Proceeding of the International Conference on Security and Management, Las Vegas, NV*, pages 310–316. Citeseer, 2003.

[61] Jyotiprakash Sahoo and Subasish Mohapatra. A Survey on Evolutionary Approaches to Intrusion Detection Systems. *rimtengg.com*, pages 1–5.

[62] Jiang Hua and Zhao Xiaofeng. Study on the Network Intrusion Detection Model Based on Genetic Neural Network. *2008 International Workshop on Modelling, Simulation and Optimization*, pages 60–64, December 2008.

[63] S.J. Stolfo, P.K. Chan, E. Eskin, M. Miller, and S. Hershkop. Real time data mining-based intrusion detection. *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, 1:89–100.

[64] JR Votano, M Parham, and LH Hall. No Title. *Chemistry &*, 2(2):16–21, 2004.

[65] Eric Bloedorn, Alan D Christiansen, William Hill, Clement Skorupka, Lisa M Talbot, Jonathan Tivel, and Dolley Madison Blvd. Data Mining for Network Intrusion Detection : How to Get Started. 1820.

[66] S Terry Brugger. Detection. 2004.

[67] I Kim, D Kim, Byoungkoo Kim, Yangseo Choi, and Seongyong Yoon. An architecture of unknown attack detection system against zero-day worm. *Proceedings of the 8th*, 2008.

[68] G Portokalidis and Asia Slowinska. Argos: an emulator for fingerprinting zero-day attacks for advertised honeypots with automatic signature generation. *ACM SIGOPS Operating Systems*, 2006.

[69] Ikkyun Kim, Daewon Kim, Yangseo Choi, Koohong Kang, Jintae Oh, and Jongsoo Jang. Validation Methods of Suspicious Network Flows for Unknown Attack Detection. 3(1):104–114, 2009.

[70] Zubair a. Baig, Saad Khan, Saif Ahmed, and Mohammed H. Sqalli. A selective parameter-based evolutionary technique for network intrusion detection. *2011 11th International Conference on Intelligent Systems Design and Applications*, pages 65–71, November 2011.

Dataset Description

KDD99 dataset is among the widely used datsets used to verify the performances of intrusion detection systems.A complete description of the dataset is provided in [52, 18]. The dataset describes each network connection on the basis of 41 parameters. These include parameters are based on the following

1. Intrinsic parameters: These are the basic parameters and are directly extracted from the headers. Table 1 shows the intrinsic feature of the dataset.

No.	Feature name	Description	Type	Value
1	duration	length (number of seconds) of the connection	continuous	Integer
2	protocol type	type of the protocol, e.g. tcp, udp, etc.	discrete	Symbolic
3	service	network service on the destination, e.g., http, telnet, etc.	discrete	Symbolic
4	flag	normal or error status of the connection	discrete	Symbolic
5	source bytes	number of data bytes from source to destination	continuous	Integer
6	dest. bytes	number of data bytes from destination to source	continuous	Integer
7	land	1 if connection is from/to the same host/port; 0 otherwise	discrete	Binary
8	wrong fragment	number of 'wrong'fragments	continuous	Integer
9	urgent	number of urgent packets	continuous	Integer

Table 1: Intrinsic Features

2. Content based parameters: These parameters are extracted from the contents of the network packets and are based on domain knowledge. Table 2 lists the content based features of the dataset

No.	Feature name	Description	Type	Value
10	hot	number of **hot** indicators	continuous	Integer
11	num failed logins	number of failed login attempts	continuous	Integer
12	logged in	1 if successfully logged in; 0 otherwise	discrete	Binary
13	num compromised	number of **compromised** conditions	continuous	Integer
14	root shell	1 if root shell is obtained; 0 otherwise	discrete	Binary
15	su attempted	1 if **su root** command attempted; 0 otherwise	discrete	Binary
16	num root	number of **root** accesses	continuous	Integer
17	num file creations	number of file creation operations	continuous	Integer
18	num shells	number of shell prompts	continuous	Integer
19	num access files	number of operations on access control files	continuous	Integer
20	num outbound cmds	number of outbound commands in an ftp session	continuous	Integer
21	is hot login	1 if the login belongs to the **hot** list; 0 otherwise	discrete	Binary
22	is guest login	1 if the login is a **guest login**; 0 otherwise	discrete	Binary

Table 2: Content based Features

3. Traffic based parameters: These parameters are calculated by considered a number previous connections as well. This category can be further classified into two sub groups.

   (a) Time traffic features: The parameters are defined based on a 2 minute time window. Table 3 show time based traffic features.

   (b) Machine traffic features: These parameters are defined based on the number of connections to the same machine. Table 4 show time based traffic

features.

No.	Feature name	Description	Type	Value
23	count	no. of connections to the same host as the current connection in the past two seconds (same host)	continuous	Integer
24	serror rate	% of connections that have SYN errors	continuous	Float
25	rerror rate	% of connections that have REJ errors	continuous	Float
26	same srv rate	% of connections to the same service	continuous	Float
27	diff srv rate	% of connections to different services	continuous	Integer
28	srv count	number of connections to the same service as the current connection in the past two seconds	continuous	Integer
29	srv serror rate	% of connections that have SYN errors	continuous	Float
30	srv rerror rate	% of connections that have REJ errors	continuous	Float
31	srv diff host rate	% of connections to different hosts	continuous	Float

Table 3: Traffic(time) based Features

4. Class: Class indicates the type of the connection. Normal indicates that the connection is of normal traffic. Any other value indicates that the connection is anomalous. The KDD99 dataset comprises of four classes of attacks namely denial of servic, probe, r2l and u2r.

No.	Feature name	Description	Type	Value
32	dst host count	No. of connections to the same destination IP address	Continuous	Integer
33	dst host srv count	No. of connections to the same destination port	Continuous	Integer
34	dst host same srv rate	% of connections that were to the same service, among the connections aggregated in dst host count (32)	Continuous	Float
35	dst host diff srv rate	% of connections that were to different services, among the connections aggregated in dst host count (32)	Continuous	Float
36	dst host same src port rate	% of of connections that were to the same source port, among the connections aggregated in dst host srv count (33)	Continuous	Float
37	dst host srv diff host rate	% of connections that were to different destination machines, among the connections aggregated in dst host srv count (33)	Continuous	Float
38	dst host serror rate	% of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst host count (32)	Continuous	Float
39	dst host srv serror rate	% of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst host srv count (33)	Continuous	Float
40	dst host rerror rate	% of connections that have activated the flag (4) REJ, among the connections aggregated in dst host count (32)	Continuous	Float
41	dst host srv error rate	% of connections that have activated the flag (4) REJ, among the connections aggregated in dst host srv count (33)	Continuous	Float

Table 4: Traffic(Machine) based Features

Vitae

- Saad Ahmed Khan.
- Date of Birth: 27-Dec-1985.
- Present Address: Al Khobar, Saudi Arabia.
- Permanent Address: S.F. Colony Karachi, Pakistan.
- Mobile Number: +966-55-6227062
- Email Address: saadahkh@hotmail.com
- Received B.S. degree in Telecommunication Engineering from National University of Computer and Emerging Sciences (NUCES-FAST) from Karachi, Pakistan in 2007
- Completed M.S. degree requirements at KFUPM, Saudi Arabia in 2013