INTRUSION DETECTION SYSTEM USING GENETIC ALGORITHM

A PROJECT REPORT

Submitted by

…….
…….

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

……
……

…….

BONAFIDE CERTIFICATE

Certified that this project report **"INTRUSION DETECTION SYSTEM USING GENETIC ALGORITHM"** is the bonafide work of **……..** who carried out the project work under my supervision.

ABSTRACT

Network security has played a major role in any network design in recent times. To provide Network security, various tools are available like firewalls and intrusion detection systems (IDS). In firewalls we can't detect *inside intruders* whereas *IDS* can detect them. Here an intrusion detection system based on *Genetic Algorithm* has been proposed. The proposed system performs its tasks at *Host level* and *Network level.* In Host level, the entire system is monitored based on the various host parameters. The user can also protect his files using *"Modified DES"* which varies from DES in the key generation part. In the network level, the system works by monitoring the packets transferred across the network. First, entire knowledge about

the network is obtained. Then using the *"cross-over"* operator of Genetic Algorithm, the system creates initial population of all sorts of combination of packets. After initial population, the system obtains the intrusion condition, priorities of the parameters and intrusion level. Then the *rule sets* are framed from the initial population based on those entries whose fitness value exceeds the intrusion level. Then based on the *rule sets* the packets are monitored. In the proposed system, the user can protect his files even from the super user; the priorities of intrusion parameters and the intrusion level can be varied according to administrator's preference.

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

INTRODUCTION

Network Security has turned out to be a more complicated and challenging area in now a day's network world. When we think of designing a network a key issue to be taken into account is preventing it from the intruders. Intruders may be classified as *inside* and *outside* intruders. *Inside intruders* who belong to the same corporation, access the files of other persons by cracking that person's password, which leads to a heavy loss in network security. *Outside* intruders are those who don't belong to the corporation but they somehow try to access the important files of the corporation. Apart, from the general classification of the intruders, we have three more classes of intruder's classification namely masquerader, misfeasor and clandestine user.

Masquerader is an individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account.

Misfeasors are those legitimate users who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges.

Clandestine users are those who seize supervisory control of the system and use this control to evade auditing and access controls or to suppress audit collection.

Some of the examples of intrusion attempts are

Attempts to copy the *password file* at a rate exceeding once every other day.

Suspicious remote procedure call (RPC) request at a rate exceeding once per week.

Attempts to connect to non-existent "bait" machines at least every two weeks.

So we have to prevent this unauthorized access and increase the network security. To do so we have various tools available like firewalls, Intrusion Detection Systems (IDS).

Firewalls generally don't detect the inside intruders because of which we go for the intrusion detection system. These system works based on the predefined set of rules, which are set by the network administrator.

1.1) EXISTING SYSTEM

Firewalls:

A firewall is a program that filters the information coming through the Internet connection into a private network or computer system. Firewalls act as a barrier between corporate (internal) networks and the outside world (Internet), and filter incoming traffic according to a security policy, thus erecting an outer security wall or perimeter between the premises network and the Internet. Firewalls are usually configured by adding "rules "that allow specific types of traffic to go through the firewall. If the filters flag an incoming packet of information, it is not allowed through. A firewall defines a single choke point that keeps unauthorized users out of protected network, prohibits potentially vulnerable services from entering the network. The firewall may be a single computer system or a set of two or more systems that co-operate to perform the firewall function.

Intrusion Detection Systems (IDS)

All intrusion detection systems address the problems of intrusion based on the following key ideas:

- If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised.
- An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.
- Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

The intrusion detection systems are based on the simple fact that the typical behavior of the intruder differs from that of a legitimate user in ways that can be quantified.

Although the typical behavior of an intruder differs from the typical behavior of an authorized user, there is an overlap in these behaviors. Thus a loose interpretation of intruder behavior, which will catch more intruders, will also lead to a number of "false positives ", or authorized users identified as intruders .On the other hand, an attempt to limit false positives by a tight interpretation of intruder behavior will lead to an increase in false negatives, or intruders not identified as intruders. Thus there is an element of compromise and art in the practice of intrusion detection.

The existing system is based on patterns of legitimate user behavior and can be established by observing past history. The following approaches to intrusion detection have been observed.

Statistical anomaly detection: Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behavior. We have two types of approaches in this system.

Threshold detection: This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.

Profile-based: A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.

Rule-based detection: Involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.

Anomaly detection: Rules are developed to detect deviation from previous usage patterns.

Penetration identification: An expert system approach that searches for suspicious behaviors.

Statistical approaches attempt to define normal, or expected, behavior, whereas rule-based approaches attempt to define proper behavior.

One of the most traditional ways of handling or detecting the presence of intruders is the usage of *audit records.*

Audit records:

A fundamental tool for intrusion detection is the audit record. Some record of ongoing activity by users must be maintained as input to an intrusion detection system. Basically, two plans are used:

Native audit records: All multi-user operating system include accounting software that collects information on user activity. The advantage of using this information is that no additional collection software is needed. The disadvantage is that the native audit records may not contain the needed information or may not contain it in a convenient form.

Detection-specific audit records: A collection facility can be implemented that generates audit records containing only that information required by the intrusion detection system. One advantage of such an approach is that it could be made vendor independent and ported to a variety of systems. The disadvantage is the extra overhead involved in having, in effect, two accounting packages running on a machine.

Most user operations are made up of a number of elementary actions. For example, a file copy involves the execution of the user command, which includes doing access validation and setting up the copy, plus the read from one file, plus the write to another file. The decomposition of a user operation into elementary actions has three advantages:

Because objects are the most important entities in a system, the use of elementary actions enables an audit of all behavior affecting an object. Thus the system can detect attempted subversions of access controls and can detect successful subversions by noting an abnormally in the set of objects accessible to the subject.

Single-object, single actions audit records simplify the model and the implementation.

Because of the simple, uniform structure of the detection-specific audit records, it may be relatively easy to obtain this information or at least part of it

by a straight forward mapping from existing native audit records to the detection-specific audit records.

Statistical anomaly detection

Statistical anomaly detection techniques as mentioned earlier fall in to two categories; they are ***threshold detection*** and ***profile-based systems*****.**

Threshold detection involves counting the number of occurrences of a specific event type over an interval of time. If the count surpasses what is considered a reasonable number that one might expect to occur, then intrusion is assumed.

Profile-based anomaly detection focuses on characterizing the past behaviors of individual users or related groups of users and then detecting significant deviation. A profile may consist of set of parameters, so that deviation on just a single parameter may not be sufficient in itself to signal an alert.

The foundation of this approach is an analysis of audit records. The audit records provide the inputs to the intrusion detection function in two ways.

First, the designer must decide on a number of quantitative metrics that can be used to measure user behaviors. An analysis of audit records over a period of time can be used to determine the activity profile of the average user. Thus, the audit records serve to define typical behaviors.

Second, current audit records are the input used to detect intrusion. That is, the intrusion detection model analyses incoming audit records to determine deviations from average behavior.

Rule based Intrusion detection:

Rule –based techniques detect intrusion by observing events in the systems and applying a set of rules that lead to a decision regarding whether a given pattern of activity is or is not suspicious.

Rule-based anomaly detection is similar in terms of its approach and strengths to statistical anomaly detection. With the rule –based approach, historical audit records are analyzed to identify usage patterns and to generate automatically rules that describe those patterns. Rules may represent past behaviors patterns of users, programs, privileges, time slots, terminals, and so on. Current behavior is then observed, and each transaction is matched against the set of rules to determine if it conforms to any historically observed pattern of behavior.

As with statistical anomaly detection, rule- based anomaly detection does not require knowledge of security vulnerabilities within the system. Rather, the scheme is based on observing past behavior and, in effect, assuming that the future will be like the past. In order for this approach to be effective, a rather larger database of rules will be needed. For example, most of the schemes contain anywhere from 10^4 to 10^6 rules.

Rule based penetration identification takes a very different approach to intrusion detection, one based on expert system technology. The key feature of such system is the rules for identifying known penetrations or penetrations that would exploit known weaknesses. Rules can also be defined that identify suspicious behavior, even when the behavior is within the bounds of established patterns of usage. Typically, the rules in these systems are specific to the machine and operating systems also; such rules are generated by "experts" rather than by means of an automated analysis of audit records. The normal procedure is to interview system administrators and security analysts to collect a suite of known penetration scenarios and key events that threaten the security of the target system. Thus the strength of the approach depends on the skill of those involved in setting up the rules.

1.2) DRAWBACKS OF EXISTING SYSTEM

The task of detecting a *misfeasor* (legitimate user performing in an unauthorized fashion) is more difficult, in that the distinction between abnormal and normal behavior may be small. Such violations would be undetectable solely through the search for anomalous behavior. However, *misfeasor* behavior might be detectable by intelligent definition of the class of conditions that suggest unauthorized use. The detection of the *clandestine* user is beyond the scope of purely automated system.

In terms of the type of attackers listed earlier, statistical anomaly detection is not effective against the misfeasors. The *threshold analysis*, by itself is and ineffective detector of even modified sophisticated attacks. Because both the threshold and time interval must be determined, which may be variable across users, such thresholds are likely to generate either a lot of false negatives or a lot of false positives.

The *profile-based system* of the anomaly detection technique has a drawback that the deviation of a single parameter may not be sufficient in itself to signal the presence of an intruder. The *profile-based* system also demands the tracking of past behavior of individual users or related group over a period of time and this involves a lot of cost and resources.

The *rule-based penetration* detection system involves the usage of rules for identifying penetrations in to the system. Again, its greatest drawback is that it involves rules set up by interviewing various experts, system administrators and security analysts to collect a suite of known penetration scenarios and key events that threaten the security of the system. This system's biggest weakness is that its security greatly involves or depends on the skills of those who are involved in setting up the rules for the system. Moreover, the system has a big set back in that it is very much specific to the machine and operating system on which it has been built.

Although the single system intrusion detection systems have been very prominent over the years the typical organization, however, needs to defend a distributed collection of hosts supported by a LAN or inter-network. The process of building up an intrusion detection system across the LAN involves a lot of co-operation and co-ordination among the various individual systems that are part of the LAN. The main drawbacks of the existing distributed detection systems are

> All existing systems face the possible challenge of having to deal with different record formats. In a heterogeneous environment, different systems will employ different native audit collection systems and, handling them requires a thorough knowledge of all the systems and their compatibilities.

Security also play a major role in such systems and involves running a secured channel across the network .One or more nodes in the network will serve as collection and analysis points for the data from the systems on the network. Thus, either raw audit data or summary data must be transmitted across the network. Therefore, there is a requirement to assure the integrity and confidentiality of these data. Integrity is required to prevent an intruder from masking his or her activities by altering the transmitted audit information. Confidentiality is required because the transmitted audit information could be valuable.

Both a centralized or decentralized architectures have been used. The centralized architecture involves a single central point of collection and analysis of all data. This eases the task of correlating incoming reports but creates a potential bottleneck in terms of resource usage and a single point of failure for the whole system. With a decentralized architecture, there are more than one analysis centers, but these must co-ordinate their activities and exchange information constantly.

1.3) NEED FOR NEW SYSTEM:

Intrusion detection is needed in today's computing environment because it is impossible to keep pace with the current and potential threats and vulnerabilities in our computing systems. The environment is constantly evolving and changing fueled by new technology and the Internet. To make matters worse, threats and vulnerabilities in this environment are also constantly evolving. Intrusion detection systems are tools to assist in managing threats and vulnerabilities in this changing environment.

Threats are people or groups who have the potential to compromise your computer system. These may be a curious teenager, a disgruntled employee, or espionage from a rival company or a foreign government .The hacker has become a nemesis to many companies.

Vulnerabilities are weaknesses in the system. Vulnerabilities can be exploited and used to compromise your system. New vulnerabilities are discovered all of the time. Every new technology, product, or system brings with it a new generation of bugs and unintended conflicts or flaws. Also the possible impact from exploiting these vulnerabilities is constantly evolving. In a worse case scenario, an intrusion may cause production downtime, sabotage of critical information, theft of confidential information, or other assets, or even negative public relations that may affect an organization.

Intrusion detection systems are tools that can assist in protecting a company from intrusion by expanding the options available to manage the risk from threats and vulnerabilities. The tool could be used to detect an intruder, and stop the exploit from use by future intruders. Intrusion detection systems can become a very powerful tool in an organization‘s security infrastructure.

1.4) PROPOSED SYSTEM:

A new intrusion detection system is been proposed, which uses Genetic Algorithms as a tool to detect the intruders. The proposed system undergoes its detection process with two levels. They are:

Host level Detection

Network level Detection.

In host-based detection, the individual systems are monitored based on the various predefined constraints, which are set by the network administrator and by the normal user, to protect their data.

In network level detection, the packets transferred in the entire network are evaluated based on the rule sets generated and the intruder’s attempts are detected. The system generally works on the lines of expert system wherein, it performs an action if a condition is valid.

If {condition} then {act}

The overall working of the system ends with the transfer of message about the details of intruders to the server from the detected system. To assist this, the system has a server and client part .The system maintains two log files, one in server, which maintains details about the intrusion attempts and the other in client about the file accessed by a particular user.

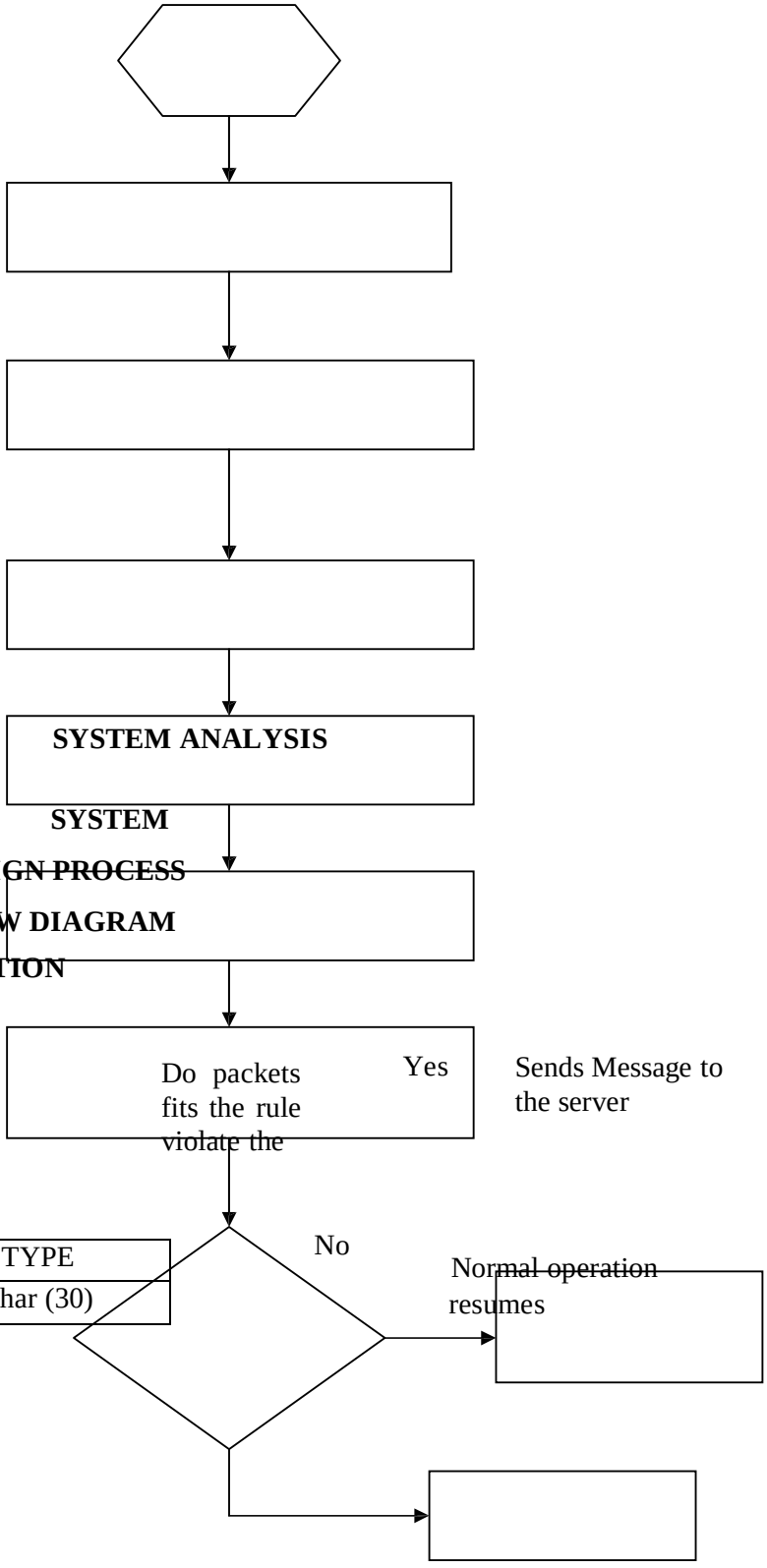


TABLE STRUCTURE

IP Address:

FIELD	TYPE
IP Address	Char (30)

Protocol:

FIELD	TYPE
Protocol Number	Integer
Protocol Name	Char (30)

Port:

FIELD	TYPE
Port Number	Integer
Port Name	Char (30)

Resource Utilization:

FIELD	TYPE
CPU Utilization	Float
Memory Utilization	Float

Banned Users:

FIELD	TYPE
User Name	Char (30)
Hostname	Char (30)

Initial Population:

FIELD	TYPE
Source IP	Char (30)
Destination IP	Char (30)
Protocol	Char (30)
Source port	Integer
Destination Port	Integer
Transmitted bytes	Big Integer
Received Integer	Big Integer
Time	Time

IDS Condition:

FIELD	TYPE
Rule Number	Char (30)
Source IP	Char (30)
Destination IP	Char (30)
Protocol	Char (30)
Source port	Char (30)
Destination Port	Char (30)

Transmitted bytes	Char (30)
Received Integer	Char (30)
Time	Char (30)

Priority:

FIELD	TYPE
Rule Number	Integer
Source IP	Integer
Destination IP	Integer
Protocol	Integer
Source port	Integer
Destination Port	Integer
Transmitted bytes	Integer
Received Integer	Integer
Time	Integer

Rule sets:

FIELD	TYPE
Rule Number	Integer
Source IP	Char (30)
Destination IP	Char (30)
Protocol	Char (30)
Source port	Integer
Destination Port	Integer
Transmitted bytes	Big Integer
Received Integer	Big Integer
Time	Time

3.4) PROGRAM MODULES

The program modules can be divided into two major classifications. They are:

Host level detection.

Network level detection.

The proposed system has a server and client part. The server part runs in the main server of the network and the client part run on all individual systems connected to the network.

In the former division, the individual systems are monitored continuously for intruders. While, in the later part, the system checks for intruders by monitoring the packets transferred in the entire network.

3.4.1) Host Based Detection

In host level detection, the IDS continuously monitors each user's system utilization and tracks down intruders based on a few threshold limits for the resources being utilized by each user. The host level detection performs its actions based on the following criteria's:

File Access.

Sensitive command usage.

Resource Utilization.

Logins and logouts.

File Access

Here the IDS applies the ***"Modified DES***" to the particular file which is specified by the user. The user can use this system and protect his file, from other users (Even from root user), by encrypting it using ***"Modified DES"*** *(M-DES)*.

The M-Des follows the same Encryption and Decryption Formula as it is in Data Encryption Standards (DES), but it varies from DES in the Key Generation Part.

Modified DES

M-Des uses three keys, two from the user and one system generated.

The Key Generation part which varies from Des is been described below:

First the user enters the key to the system, which will be converted to a 64 bit. From which we get the value of the key (Key 1).

User KEY = "Network" = 64 bits value (A) = Bit Value.

That value is given to the pseudo random number generator Equation to obtain the system generated key, which is given to the user (Key 2).

Bit value = Equation = System Key (B) = user.

Than XOR User & system key and the data is encrypted using this key.

(B) XOR (A) = new key = DES.

Split new key to two 32 bits. (X & Y).

From the two values, the varying bits are given to the user as cross-over points, out of which user selects one (Key 3).

Compare X & Y for varying bit = user = Cross point.

From the user given cross over point we perform a crossover on X & Y.

From cross point given by the user get 48 bits and split it into two (U & V).

In the first 24 bit (U) perform the 16bit shift.

In second part (V) perform mutation

Perform last two steps for 16 rounds with operations on U & V exchanged every time.

A log file is maintained which maintains the details of the users who accessed the file. And if a user is going to attempt to decrypt the files more than the threshold level, a message is sent to log file.

Sensitive commands

Here, the system requests the network administrator to specify a few sensitive commands for which access rights is not allowed for the normal unauthorized users. Whenever the user tries to use any of these commands, the client system indicates to the server system that a user is trying to use the sensitive commands and the server adds this user's in its log file as an intruder's attempt.

Resource utilization

Here, the system constantly checks the various client systems' resource utilization like Central Processing Unit (CPU) utilization, Memory Utilization e.t.c. Once, this exceeds the particular threshold level, the system indicates to the server about the particular user who is using the resource. This message is then sent to the server, which adds this particular user in its log file.

Logins and logouts:

Here the system performs its intrusion detection by taking the following actions:

Frequent Logins and Logouts.

Multi system logging.

Banned user.

Frequent Logins and Logouts

Here, the system checks for the presence of intruders by comparing the frequency of logins to a threshold level and once it exceeds the threshold level, it concludes the user as an intruder.

Multi system logging.

When there are many numbers of simultaneous logins with the same user name, the system concludes that the user is an intruder.

Banned user.

The network administrator can use this property to ban any users logging in. The IDS continuously monitors the system for those users who are banned by the network administrator.

In the above cases a message is sent to the server that the above user is an intruder and the server adds this user as an intruder in its log file.

3.4.2) Network Level Detection.

In network detection, the system monitors the packets transferred in the network for intrusion detection. The system considers following parameters for performing its action:

Destination IP Address.

Source IP Address.

Protocol.

Source Port Number.

Destination Port Number.

Number of bytes transferred.

Number of bytes Received.

Time.

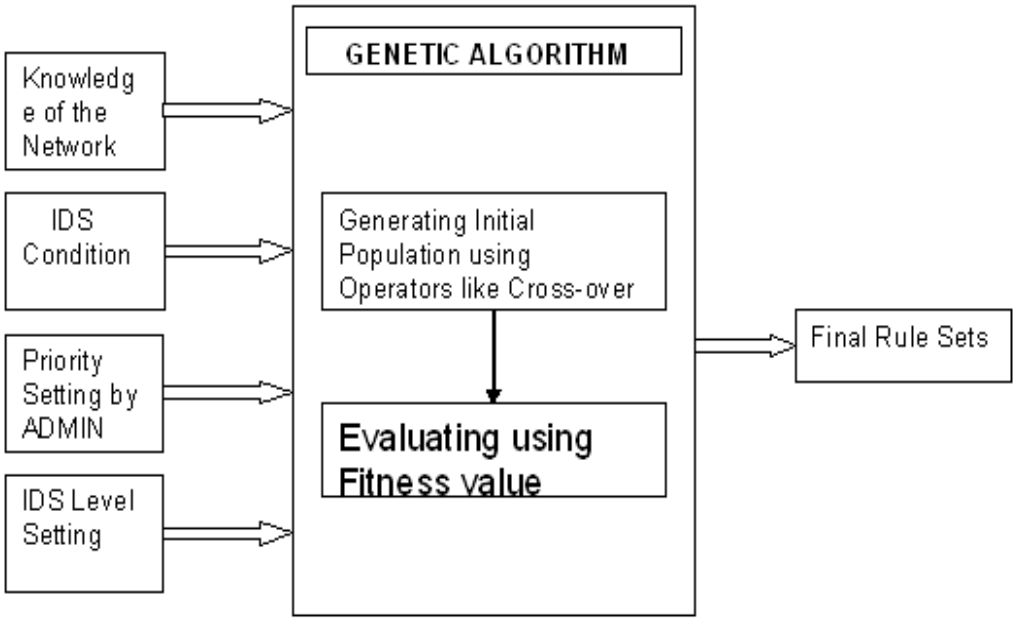


Figure 2: Overall Working of Network level detection

The above figure shows the overall working of the system at network level. The system undergoes following steps in intrusion detection:

Knowledge Collection.

Initial Population Generation.

IDS Condition Retrieval.

Setting Priorities for IDS Parameter.

Fitness value Calculation.

IDS Level Retrieval.

Rule sets Creation.

Distribution of Rule sets to the clients.

Monitoring packets for Intrusion detection.

The forth-coming section deals with the discussion on each step discussed above:

Knowledge Collection

In this initial step, the entire knowledge about the network is obtained from the network administrator. The knowledge parameters include Destination IP address, Source IP Address, Protocols, e.t.c.,

Initial population Generations

In this step, using the knowledge collected from previous step, we generate all possible combinations of the values, which form the initial population for the rule set. This is been done using the Genetic operator known as cross-over.

IDS Condition Retrieval.

In this step, we collect the ids conditions from the network administrator. The network administrator may take the parameters, which he prefers according to his IDS condition.

Setting Priorities for the IDS parameters.

In this step, the priorities of the parameter in the IDS condition are retrieved. The network admin can vary the priorities of the parameters according to his requirements. i.e., he can fix a high priority for those parameters which need maximum protection.

Fitness Value Calculation.

After the retrieval of IDS condition and priorities for them, the next step is to calculate the maximum fitness value for the condition. This is done using the formula:

Fitness = sum of (priority * parameter Value)

IDS Level Retrieval

After the fitness value calculation the value is send to the user and the Network administrator selects the IDS Level. Accordingly, the system will adapt to that intrusion level.

Rule sets Creation

After IDS level retrieval, the value is sent to the IDS Engine. Now the IDS Engine will process the Initial population sets and selects the one, which has its fitness value greater or equal to the IDS level. This is done using the priorities assigned before. Now these selected sets will form the Rule sets for intrusion detection.

Distribution of Rule sets

After the rule sets are created, the rule sets are transferred to all the clients in the network. Now the clients will monitor the packets with this rule sets for intrusion.

Packet Monitoring Process

This part of the system involves capturing the packets from all the physical segments connected in a process known as packet sniffing. Once the packets are captured, the IDS prepares for faster processing of the packets by analyzing only the most important fields in a packet. Once this is done, the system compares the packets with the rule sets generated previously. From the results of the above process the attack by an intruder is detected. Once, the presence of an attack is detected, a message is sent to the Server system, which adds this user's name in its log file as yet another intruder.

SYSTEM TESTING

4.1) Testing in Host Level Detection

In host level detection, testing is been applied for registering banned users and setting threshold level for resource utilization.

INPUT : Enter the user name and host name for the banned users.

OUTPUT: If the user and host name exists the system registers the value. Else it returns error.

Next we test the working of File access using M-DES.

INPUT : User enters the file name & passwords for the same.

OUTPUT: If file exists the system continues its encryption or decryption process, else it returns that no such file exists with that name.

4.2) Testing in Network Level Detection

In network level detection, testing is applied in knowledge collection.

INPUT : Administrator enters the IP address of client, protocol and port.

OUTPUT: If the administrator enters the IP of the client in correct system, the system registers the client and updates it in Initial population else it returns error.

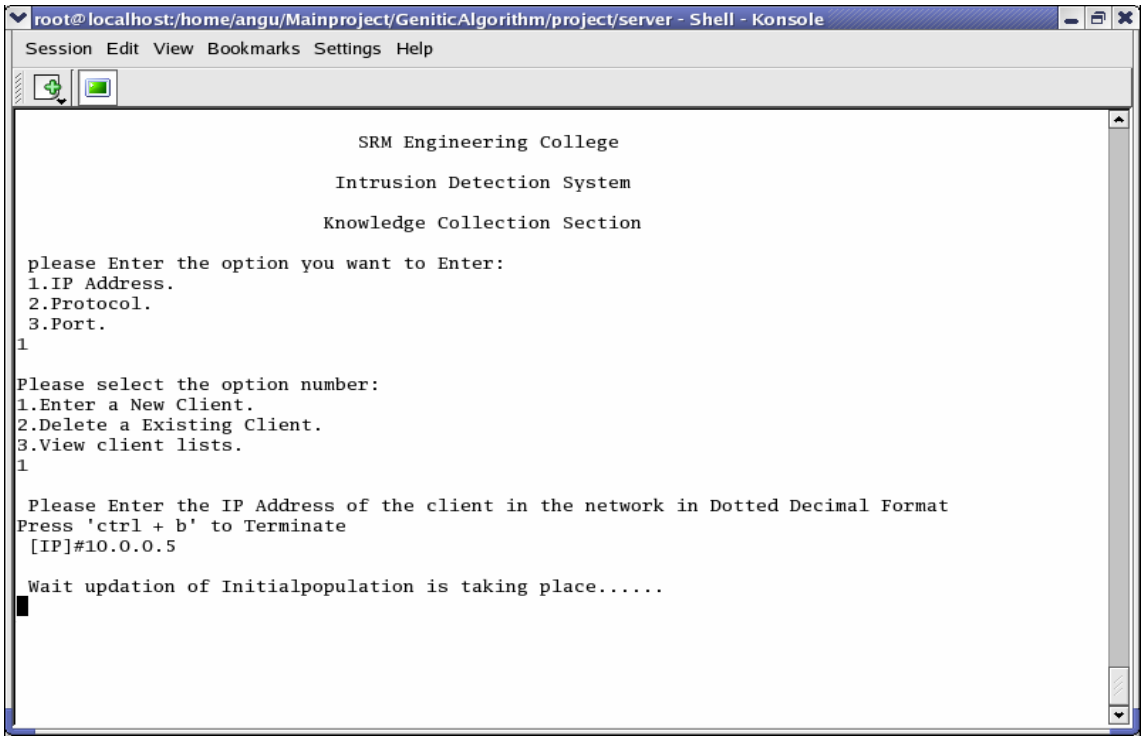
Next Testing is in IDS condition Retrieval and priority setting of the condition.

INPUT : Administrator enters IDS condition & the priority for the same

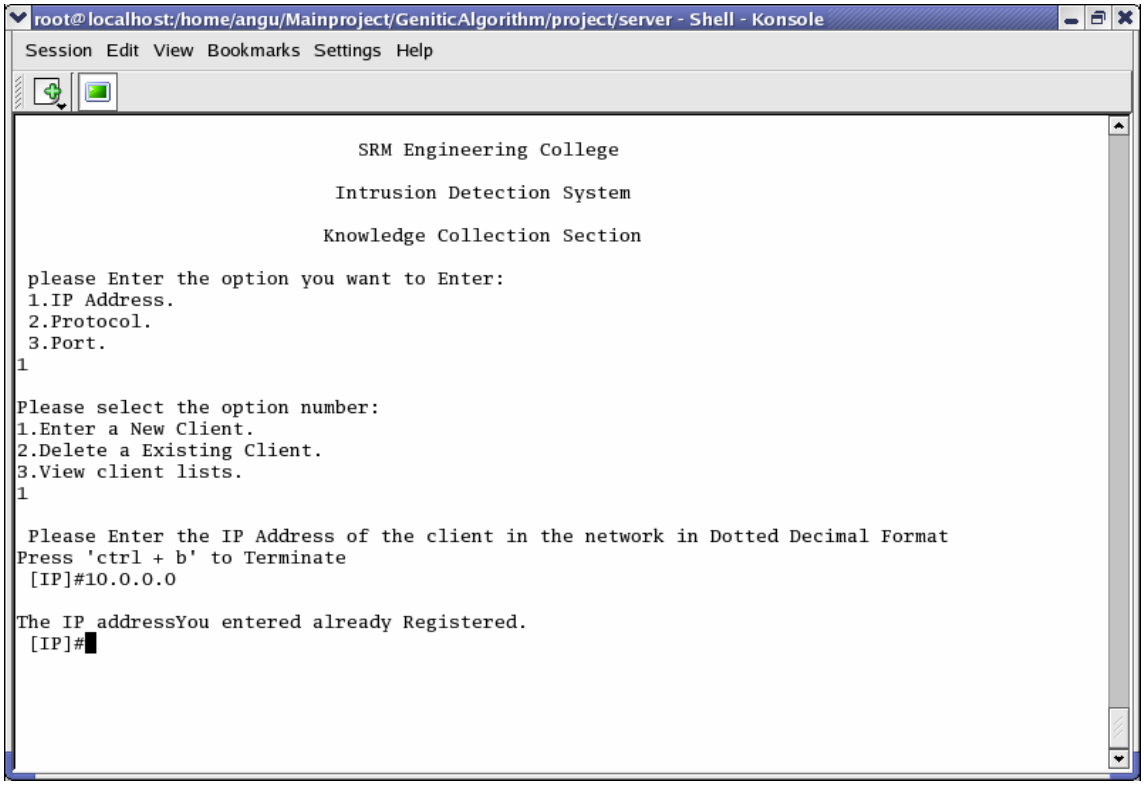
OUTPUT: If the administrator enters the IP of the registered client, protocol e.t.c., The system generates the rule sets for them .Other wise it returns an error message. Similarly if priority values vary from the prescribed range, the system throws an error.

RESULT OF TEST CASES

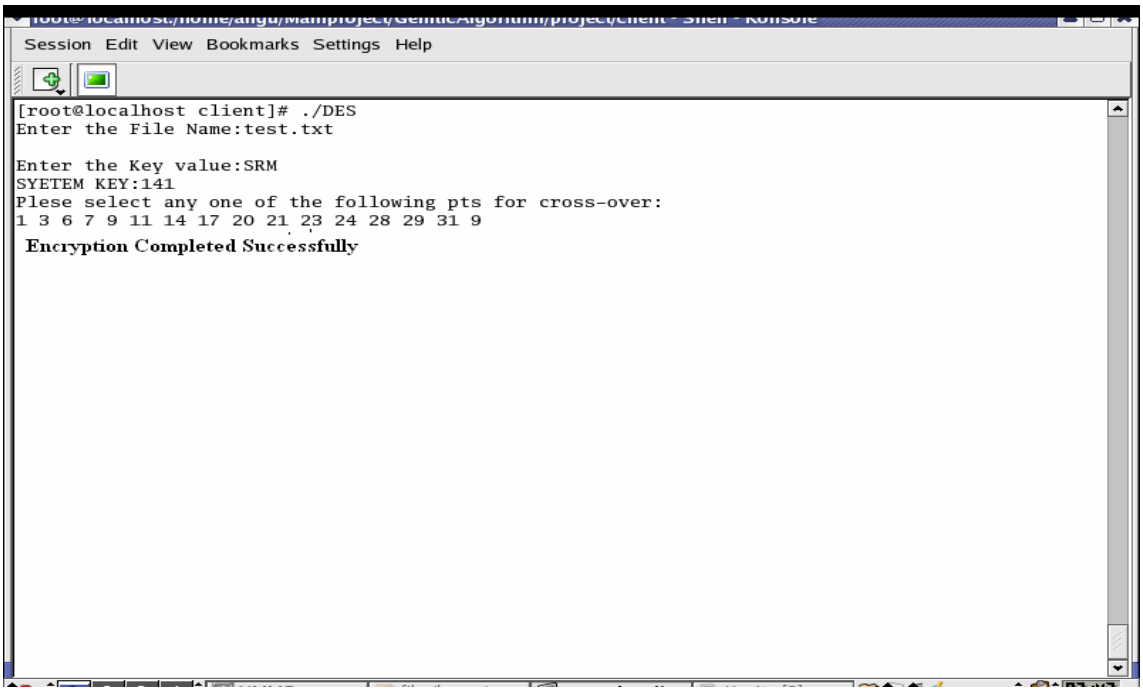
Knowledge Collection: Successful Client Registration



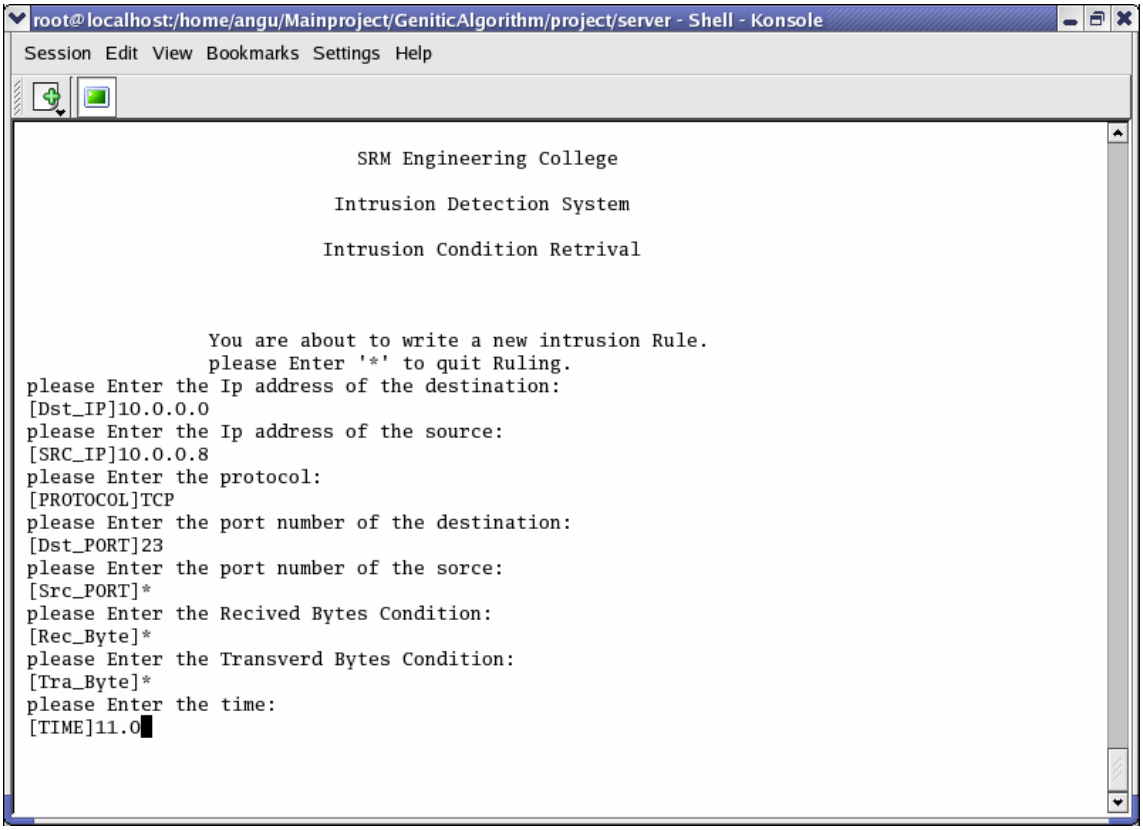
Knowledge Collection: Error in Client Registration



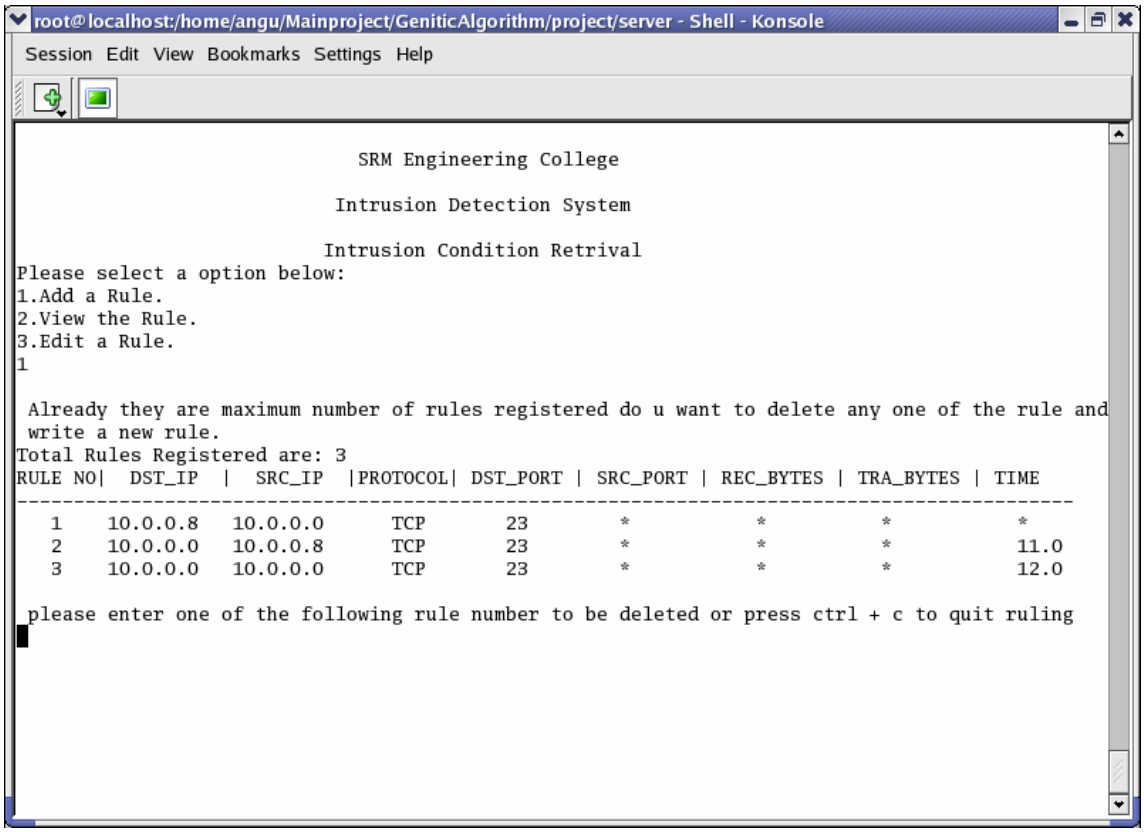
M-DES Encryption

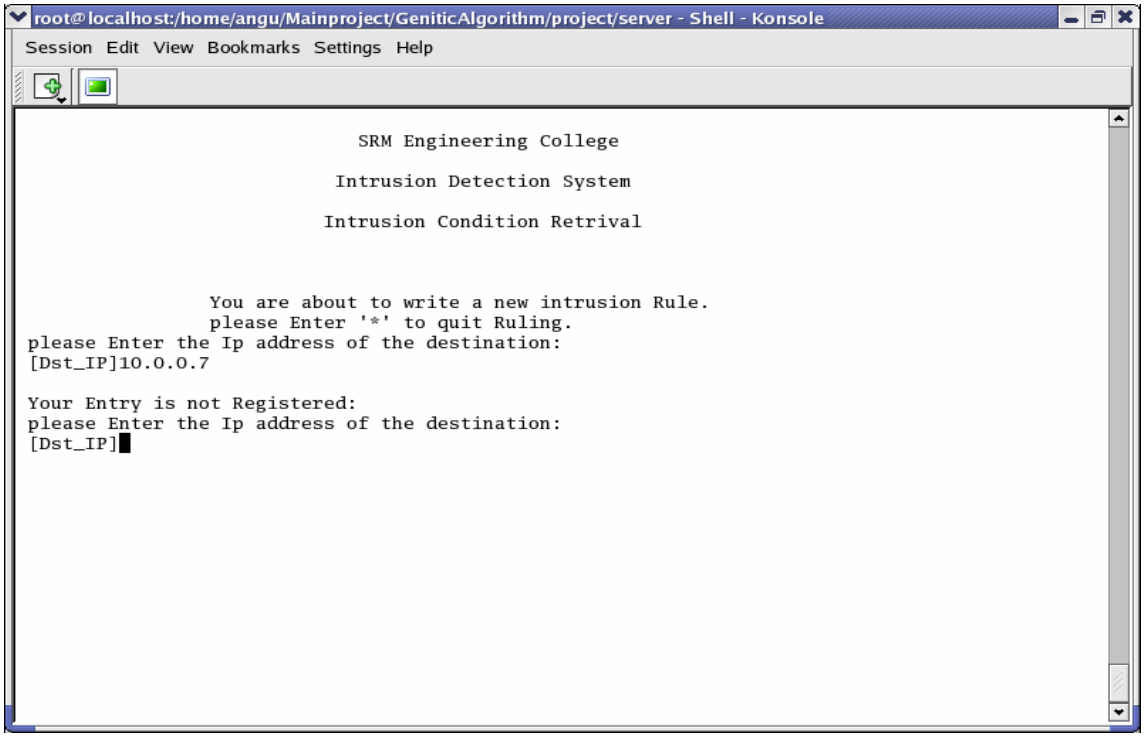


IDS Condition Retrieval

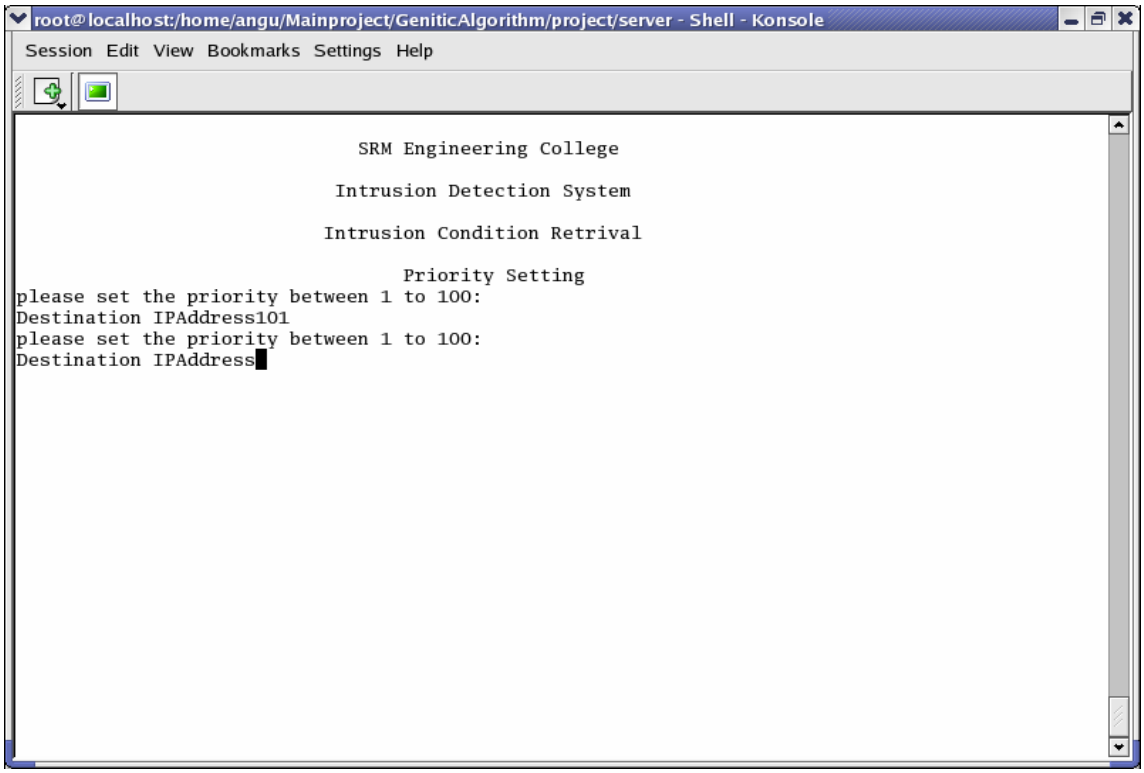


IDS Condition Retrieval Error Messages

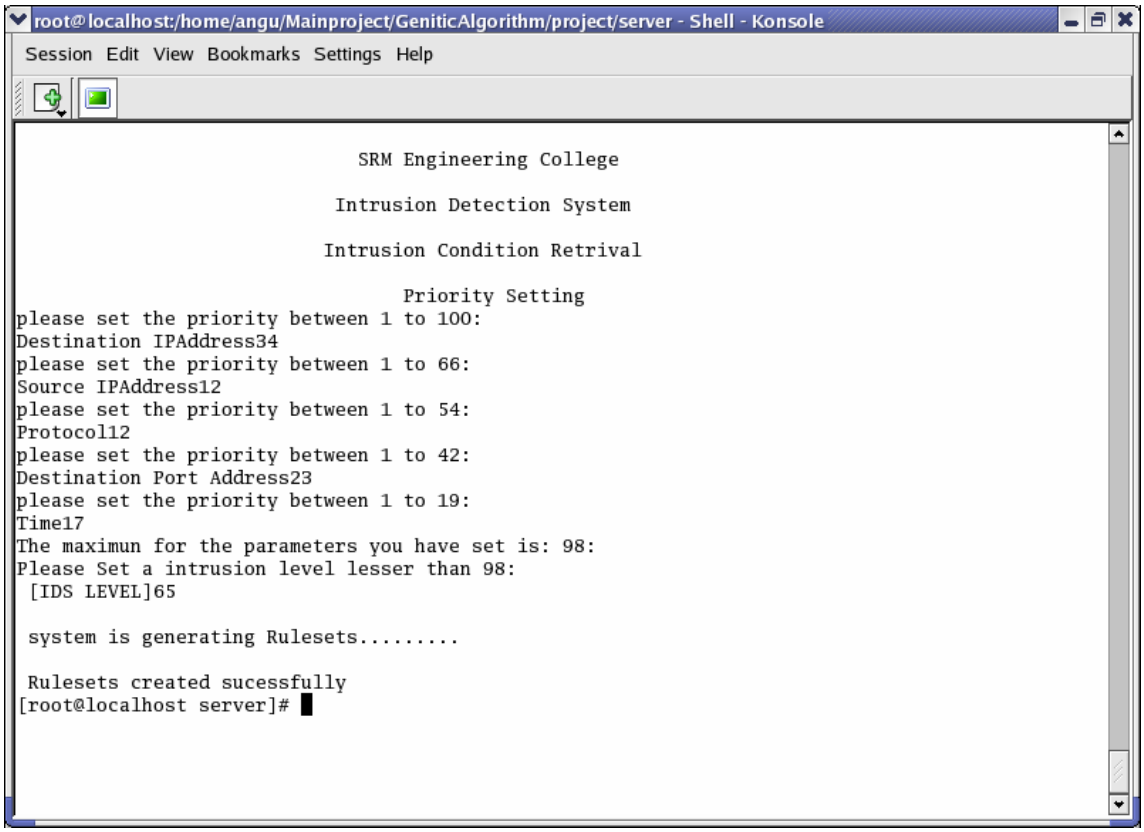




Priority Setting: Error Message



Rule set Creation:



CONCLUSION

Thus using the IDS, the network administrator can vary the intrusion level of the system according to the needs of the security for the network. The system also allows the network administrator to vary the priorities of the various parameters of intrusion. The user also has the advantage of protecting the files in any form by using "Modified DES ".

Advantages:

Since both host-based and network-based are used, the system can handle intruders both at the standalone system level and the network level, which has been a big challenge off late.

Any user can protect his files from any other user (including super user).

Network administrator can vary the priority & intrusion level according to his needs.

The IDS uses logs containing the various events and hence they can measure whether an attack was successful or not with very good efficiency.

Future enhancements

To develop a system in which the user can protect his files from copying to other user space.

To develop a system in which the user can make his file read only (Even for Root).

To develop a system, which can detect intruders, using Honey pots.

To develop a system, that can detect outside intruders.

APPENDIX -1

SAMPLE CODE:

MODIFIED DES:

/*MDES.h – A cryptographic API

***** MDES.h performs the process of Encryption and Decryption according to *DES but differs in the key generation part. This file provides the *functionality of encrypting a file. Using this file any user can encrypt the *file and protect the same from any other user or super user. The MDES *uses three keys, out of which two is provided by the

user and one is *generated by the system based on the keys provided by the user. The *MDES uses Genetic algorithm's operators like cross over and mutation for *Key generation. The first key of the user can be of any type and the next *one is the cross over point**/**

```
/*****************************MDES.h**********************/
#include<stdio.h>     /* Standard Input/ output library */
#include<math.h>   /* Mathematical operators*/
#include<string.h>  /* String manipulations*
/*****************************************************************/
#define NAME 1000
#define Max 100000
/*****************************************************************/
                    /* Common declarations */
          int bin[70],bin_ip[64],sykey[17][64];
          int key[16][48];
          int perm[32],syskey;
          FILE *fp,*fp1;
          char filename[NAME];
          char Key[Max];
          int binary[Max],skey[Max];
          int keylmt = 0;
/*****************************************************************/
/* Following are the general format of permutation according to DES*/
                              /* Initial Permutation */
          int IP[64] =  {57,49,41,33,25,17,9,1, 59,51,43,35,27,19,11,3,
61,53,45,37,29,21,13,5, 63,55,47,39,31,23,15,7, 56,48,40,32,24,16,8,0,
58,50,42,34,26,18,10,2, 60,52,44,36,28,20,12,4, 62,54,46,38,30,22,14,6};
                              /* Expansion/Permutation */
          int E_P[48] =  {31,0,1,2,3,4,3,4,5,6,7,8,7,8,9,10,11,12,11,12,13,14,15,16,
15,16,17,18,19,20,       19,20,21,22,23,24, 23,24,25,26,27,28,
27,28,29,30,31,0};
                              /* S-Box Formats */
                              /* S1-Box Formats */
          int s1[4][16] = {{14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7},
```

```
                    {0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8},
                    {4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0},
                    {15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13}};
                   /* S1-Box Formats */
int s2[4][16] = {{15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10},
                    {3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5},
                    {0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15},
                    {13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9}};
                   /* S3-Box Formats */
int s3[4][16] = {{10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8},
                    {13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1},
                    {13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7},
                    {1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12}};
                   /* S4-Box Formats */
int s4[4][16] = { {7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15},
                    {13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9},
                    {10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4},
                    {3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14}};
                   /* S5-Box Formats */
int s5[4][16] = { {2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9},
                    {14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6},
                    {4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14},
                    {11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3}};
                   /* S6-Box Formats */
int s6[4][16] = {  {12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11},
                    {10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8},
                    {9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6},
                    {4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13}};\
                   /* S7-Box Formats */
int s7[4][16] = { {4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1},
                    {13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6},
                    {1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2},
                    {6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12}};
                   /* S8-Box Formats */
```

```
        int s8[4][16] = {{13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7},
                         {1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2},
                         {7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8},
                         {2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11}};
                        /* Permutation*/
        int P[32] =     {15,6,19,20,28,11,27,16, 0,14,22,25,4,17,30,9,
1,7,23,13,31,26,2,8, 18,12,29,5,21,10,3,24  };
                        /* Inverse Initial permutation */
        int IP_Inv[64] = {39,7,47,15,55,23,63,31, 38,6,46,14,54,22,62,30,
                          37,5,45,13,53,21,61,29, 36,4,44,12,52,20,60,28,
                          35,3,43,11,51,19,59,27, 34,2,42,10,50,18,58,26,
                          33,1,41,9,49,17,57,25,  32,0,40,8,48,16,56,24 };
        /* initial Phase – Getting values for Encryption*/
void Encryption(int ch[])
{
        int i=0,j=0,k=0;
        int temp_b[10];
        /* Changing char value to standard format of  64 bit value */
        for(k=0;k<8;k++)
        {
                j=0;
                if(ch[k] > -1)
                {
                        while(ch[k]>1)
                        {
                                temp_b[j] = ch[k] % 2;
                                j++;
                                ch[k] = ch[k] / 2;
                        }
                        temp_b[j] = ch[k];
                        while(j<7)
                        {
                                j++;temp_b[j] = 0;
                        }
```

```
                        while(j>=0)
                        {
                                bin[i] = temp_b[j];
                                j--;i++;
                        }}}
        while(i<64)
        {
                bin[i] = 0;
                i++;
        }
          /* Sending the 64 bit value to IP */
        init_perm( bin,bin_ip );
       /* Sending bits to rounds) */
        for( j = 0;j<16;j++)
        {
                Rounds(bin_ip,key,j);     // sending bits to 16 rounds
                switching(bin_ip,perm);  // for switching
        }
        swap(bin_ip);                   // for final swapping
        Init_perm_inv(bin_ip);          //for IP-1*/
        file(bin_ip);                  // for cipher text creation
}
/*************************************************************/
        /* Doing the Initial Permutation which is the Initial phase of DES*/
void init_perm(int bin[],int bin_ip[])
{
        int i = 0, j = 0,temp_ip[64];
                /* Setting to initial permutation */
        for(i=0;i<64;i++)
        {
                bin_ip[i] = bin[IP[i]];
                temp_ip[i] = bin_ip[i];
        }}
/*************************************************************/
```

```
/* Sending to 16 round process which does manupulations with the key values on the
data*/
void Rounds(int bin_ip[],int Key[16][],int dim)
{
	int temp[48],temp_n[32];;
	int i=0,j=0,k=0;
		/* Getting the initial 32 bits for the round process */
	for(i = 32;i<64;i++)
	{
		temp_n[j] = bin_ip[i];
		j++;
	}
	/* Setting to E/P values which is a Permutation values of DES*/
	for(i = 0;i<48;i++)
	{
		temp[i] = temp_n[E_P[i]];
	}
	/* Performing XOR on E/P value and left 32 bits*/
	for( j=0;j<48;j++)
	{
		temp[j] = xor(key[dim][j],temp[j]);
	}
	for(j=0;j<=keylmt;j++)
	{
		temp[j] = xor(sykey[dim+1][j],temp[j]);
	}
	S_box(bin_ip,temp);			// Sending to S_Box
}
/*************************************************************/
		/* Performing S box function of DES*/
void S_box(int bin_ip[],int temp[])
{
	int i=0,j=0,k=5,row,column;
		/* Getting values from S-Box */
```

```
        for( i=0;i<8;i++)
        {
                row = Bin2_to_int(temp[j],temp[k]);      // Setting row
                column = Bin4_to_int(temp,j + 1,k – 1); //Setting column
                substitution(perm,i,row,column);         //changing 48 to 32
                                                                bits
                j = j + 6;
                k = k + 6;
        }
                /* sending bits for permutation */
        Permutation(P,perm);
}
/*************************************************************/
                        /* For S- box substitution */
void substitution(int perm[],int select,int row,int column)
{
/* To get S_BOX value from the first two bit as row and last 4 bit for column */
        switch(select)
        {
                case  0:
                        int_to_bin(perm,s1[row][column],select);
                        break;
                case  1:
                        int_to_bin(perm,s2[row][column],select);
                        break;
                case  2:
                        int_to_bin(perm,s3[row][column],select);
                        break;
                case  3:
                        int_to_bin(perm,s4[row][column],select);
                        break;
                case 4:
                        int_to_bin(perm,s5[row][column],select);
                        break;
```

```
                case 5:
                        int_to_bin(perm,s6[row][column],select);
                        break;
                case 6:
                        int_to_bin(perm,s7[row][column],select);
                        break;
                case 7:
                        int_to_bin(perm,s8[row][column],select);
                        break;
                default:
                        break;
        }}
                /* To change S-Box obtained value to binary bits */
void int_to_bin(int perm[],int value,int select)
{
        int temp_1[4],i=0;
        int j = 4 * select;
        while(value>1)
        {
                temp_1[i] = value % 2;
                value = value / 2;
                i++;
        }
        temp_1[i] = value;
        while(i<3)
        {
                i++;
                temp_1[i] = 0;
        }
        for(i = 3;i>-1;i--)
        {
                perm[j] = temp_1[i];
                j++;
        }}
```

```
**********************************************/
                /* For final Permutation of DES*/
void Permutation(int P[], int perm[])
{
        int i = 0;
        int temp_p[32];
                /* Setting final permutation */
        for( i = 0;i<32;i++)
        {
                temp_p[i] = perm[P[i]];
        }
                /* For final XOR */
        for( i = 0;i<32;i++)
        {
                 perm[i] = xor(temp_p[i],bin_ip[i]);
        }}
/*************************************************************/
        /* initial swift of 32 bit shift of data after every round of DES*/
void switching(int bin_ip[],int perm [])
{
        int i =0, j = 32,k=0;
                /* Initial swift of 32 bits to next rounds */
        for(i = 0;i<32;i++)
        {
                bin_ip[i] = bin_ip[j];
                bin_ip[j] = perm[i];
                j++;
        }}
/*************************************************************/
        /* Final Swaping after 16 rounds */
void swap(int bin_ip[])
{
        int temp_s[64],i = 0,j = 32;
        for(i=0;i<64;i++)
```

```
	{
		temp_s[i] = bin_ip[i];
	}
	for(i=0;i<32;i++)
	{
		bin_ip[i] = temp_s[j];
		j++;
	}
	j = 32;
	for(i=0;i<32;i++)
	{
		bin_ip[j] = temp_s[i];
		j++;
	}}
/*************************************************************/
/* Performing the inverse initial population of DES to produce cipher text*/
void Init_perm_inv(int bin_ip[])
{
	int temp_ip1[64],i;
	for(i=0;i<64;i++)
	{
		temp_ip1[i] = bin_ip[IP_Inv[i]];
	}
	for(i=0;i<64;i++)
	{
		 bin_ip[i] = temp_ip1[i];
	}}
/*************************************************************/
	/* For Cypher text file creation of encrypted data*/
void file(int bin_ip[])
{
	static int check = 0;
	int i;
	char temp_f[100000];
```

```
        if(check == 0)
        {
                /* Getting the Extension */
                createfile(filename,temp_f);
           fp1 = fopen(temp_f,"w");
                check = 1;
        }
        for(i=0;i<64;i++)
        {
                /* Writing the 1's & 0's */
                sykey[j] = temb_b[j];
                if( bin_ip[i] == 1)
                {
                        fputc(49,fp1);
                }
                else if( bin_ip[i] == 0)
                {
                        fputc(48,fp1); // writing to the file
                }
        }
        /* getting the Extension of the given file*/
void createfile(char filename[],char temp_f[])
{
        int i = 0,j = 0;
        char temp_ext[10];
        for(i = 0;i<strlen(filename);i++)
        {
                if(filename[i] =='.')
                        {
                                break;
                        }
                else
                        {
                                temp_f[j] = filename[i];
```

```
                        j++;
                }
        }
        strcat(temp_f,"Enc"); j=0;
        for(i;i<strlen(filename);i++)
        {
                temp_ext[j] = filename[i];
                j++;
        }
        strcat(temp_f,temp_ext);
}
/*************************************************************/
        /*Key generation part of MDES*/
void keygeneration()
{
        int i=0,j=0,ascii=1,va,cc,syskey,val,lmt=0;;
        unsigned long int valu;
        printf("\nEnter the Key value:");
                i = getchar();
                get:
                i = getchar();
                if(i!=10)
                {
                        va = int_to_binary(i,binary);
                        goto get;
                }
                /* Getting binary bits of the key1*/
        KEY(binary,va);
                /* Getting values to that 64bits OF key1*/
        valu = bin_to_int(binary);
                /* Passing the value to the equation to get the system key */
        syskey = Equ(valu);
       printf("SYETEM KEY:");
```

```
        printf("%d",syskey);
                /* Appending the system key to the key value */
        lmt = key_to_bit(syskey,skey,binary);
                /* Getting the crossover point from the user */
        cc = points(binary);
                /* Performing the crossover on the key1 */
        crossover(cc,binary);
                /* getting the key for 16 Rounds */
        for(i=0;i<16;i++)
        {
                keysforrounds(sykey,lmt);
        }}
**********************************************************/
Void int_to_binary(int value,int binary[])
{
        int j=0,k=0;
        static int i = 0;
        int temp_b[32];
        /* changing char value to 64 bit value */
                        while(value>1)
                        {
                                temp_b[j] = value % 2;
                                j++;
                                value = value / 2;
                        }
                        temp_b[j] = value;
                        while(j<7)
                        {
                                j++;temp_b[j] = 0;
                        }
                        while(j>=0)
                        {
                                binary[i] = temp_b[j];
                                j--;i++;
```

```
                        }
        return (i-1);
}
**********************************************************/
        /* Getting binary bit value for key */
void KEY(int binary[],int val)
{
        int i = 0,j = 0,init = val,len = 0,times = 0,k = 0;
        val = ( (val+1) % 64 );
        len = (init + (64 – val));
        if( val != 0)
                for(i=(init + 1);i<=len;i++)
                {
                        binary[i] = 0;
                }
        times = ( (i – 1) / 64);
        for( i = 1;i<times;i++)
        {
                k = i * 64;
                for(j = 0;j<64;j++)
                {
                        binary[j] = xor(binary[j],binary[k]);
                        k++;
                }}}
                /* Changing the binary value to integer */
unsigned long int bin_to_int(int binary[])
{
        int i,j=0,len,k=0,cnt = 1;
        unsigned long int val = 0;
        for(i = 0;i<4;i++)
        {
                for(k = 0;k<16;k++)
                {
                        val = val + (binary[j] * ( power(2,k) ));
```

```
                        j++;
                }}
return val;}
                /* Getting the system key for the Key */
int Equ(unsigned long int value)
{
        double val = value/5;
        int X,Y;
        ran:
                /* Getting random numbers */
                X = (int) ( val * rand() / ( RAND_MAX + 1.0 ));
              Y = (int) ( val * rand() / ( RAND_MAX + 1.0 ));
                if( (  ( (X*X*X) + (Y*Y) ) >= (value – 10) ) && ( ( ( (X*X*X) +
(Y*Y) ) ) <= (value + 10) ) )
                {
                /* System key */
                return (X+Y);
                }
        else
                goto ran;
}
/*************************************************************
        /* Appending the system key to the original key */
int key_to_bit(int key,int skey[],int binary[])
{
        int j=0,k=0,l=0,ret;
        int i = 0;
        int temp_b[32];
        /* Changing char value to 32 bit value */
                        while(key>1)
                        {
                                temp_b[j] = key % 2;
                                j++;
                                key = key / 2;
```

```
                }
                temp_b[j] = key;
                k=l=j;
                for(i=0;i<=l;i++)
                {
                        sykey[0][i] = temp_b[k];
                        k--;
                }
                ret = (i-1);
                i=0;
                while(j<31)
                {
                        j++;temp_b[j] = 0;
                }
                while(j>=0)
                {
                        skey[i] = temp_b[j];
                        j--;i++;
                }
        j = 32;
    /* final appending the system key to last 32 bit of the key */
        for(i=0;i<32;i++)
        {
                k=j+i;
                binary[k] = xor(binary[k],skey[i]);
        }
    keylmt = ret;
    return ret;
}
/*************************************************************
    /* Getting the cross over points for performing crossover*/
int points(int binary[])
{
    int pts[64],i,j = 0,k =0,pt,l;
```

```
	for(i=0;i<32;i++)
	{
		if(binary[i] != binary[i+32])
		{
		/* getting points */
			pts[k] = i;
			k++;
		}
	}
	if(k != 0)
	{
		 printf("\nPleasse select any one of the following pts for cross-
over:\n");
		for(l=0;l<k;l++)
		{
			 printf("%d ",pts[l]);
		}
		scanf("%d",&pt);
		return pt;
	}
	if(k == 0)
	{
		scanf("%d",&pt);
		return pt;
	}
}
		/* Performing cross over of genetic algorithm*/
void crossover(int pt,int binary[])
{
	int temp,tem[48],i,j=0,k=0,init,end;
	for(i=pt;i<32;i++)
	{
		/* genitical crossover */
		temp = binary[i];
```

```
                binary[i] = binary[i*2];
                binary[i*2] = temp;
        }
        for(j=0;j<=48;j++)
        {
                /* converting key to 48 bits */
                tem[j] = binary[pt];
                pt++;
                if(pt == 64)
                        pt = 0;
        }
        for(k=0;k<48;k++)
        {
                binary[k] = tem[k];
        }}
/*************************************************************/
        /* getting 16 keys for rounds */
void keysforrounds(int sykey[17][64],int lmt)
{
        int j =0,k=0,tem,len,lng,mod;
        static int i = 1;
        mod = (i%2);
        if(mod!=0)
        {
                len = 0;
                lng = 24;
        }
        else if(mod == 0)
        {
                len = 24;
                lng = 0;
        }
                /* Doing 16 bit shift */
        for(j=len;j<(len +16);j++)
```

```
        {
                tem = binary[len];
                for(k=len;k<(len+23);k++)
                {
                        binary[k] = binary[k+1];
                }
                binary[len+23] = tem;
        }
                /* doing mutation */
        for(j = lng;j<(lng+24);j++)
        {
                if(binary[j] == 0)
                        binary[j] = 1;
                else if(binary[j] == 1)
                        binary[j] = 0;
        }
        /* Appending bits */
        for(j=0;j<48;j++)
        {
                key[i-1][j] = binary[j];
        }
        for(j=0;j<=lmt;j++)
        {
                sykey[i][j] = xor(sykey[i-1][j],key[i-1][j]);
        }
        i++;
}
**********************************************************
        /* Decryption of the cyphertext */
void Decryption(int ch[])
{
        int i=0,j=0,k=0;
        int temp_b[10];
        /* Getting 64 bits separately */
```

```
        while(i<64)
        {
                if(ch[i]==48)
                        bin[i] = 0;
                else if(ch[i]==49)
                        bin[i] = 1;
                i++;
        }
                /* Initial permutation */
        init_perm(bin,bin_ip);
        for( j = 15;j>=0;j--)
        {
                Rounds(bin_ip,key,j);  // sending bits to 16 rounds
                switching(bin_ip,perm);  // for switching
        }
        swap(bin_ip);                   // for final swaping
        Init_perm_inv(bin_ip);          // Inverse Initial permutation
        Dfile(bin_ip);                   // Creating plain text
}
/*********************************************************
        /* Creating key for decryption */
void Dkeygeneration()
{
        int i=0,j=0,ascii=1,va,cc,syskey,val,lmt=0;
        unsigned long int valu;
        printf("\nEnter the Key value:");
                i = getchar();
                get:
                i = getchar();
                if(i!=10)
                {
                        va = int_to_binary(i,binary);
                        goto get;
                }
```

```
		/* Getting key to binary*/
	KEY(binary,va);
	valu = bin_to_int(binary);
	/* Getting the system key from the user */
	printf("\nplease Enter the System key:");
	scanf("%d",&syskey);
	/* Appending the system key to the key */
	lmt = key_to_bit(syskey,skey,binary);
	/* Getting the cross over point */
	cc = points(binary);
	/* Performing cross over */
	crossover(cc,binary);
	/* Getting the key for 16 rounds */
	for(i=0;i<16;i++)
	{
		keysforrounds(sykey,lmt);
	}}
/*********************************************************/
			/* Creating plaintext */
void Dfile(int bin_ip[])
{
	static int check = 0;
	int i,j,k,values = 0;
	char temp_f[100000];
	if(check == 0)
	{
		Dcreatefile(filename,temp_f);
	      fp1 = fopen(temp_f,"w");
		check = 1;
	}
		values = 0;
		k=7;
	for(i=0;i<64;i++)
	{
```

```
                    /* Getting the ASCII value */
                values = values + (bin_ip[i] * power(2,k));
                /* Writing the values to the plain text */
                if(k == 0)
                {
                        fputc(values,fp1);
                        k=7;
                        values=0;
                }
                else
                        k--;
        }}
        /* Getting the Extension */
void Dcreatefile(char filename[],char temp_f[])
{
        int i = 0,j = 0;
        char temp_ext[10];
        for(i = 0;i<strlen(filename);i++)
        {
                if((filename[i] ==’E’)&&(filename[i+1] ==’n’)&&(filename[i+2]
==’c’)&&(filename[i+3] ==’.’))
                        {
                                break;
                        }
                else
                        {
                                temp_f[j] = filename[i];
                                j++;
                        }       }
        strcat(temp_f,”D”); j=0;
        for(i=i+3;i<strlen(filename);i++)
        {
                temp_ext[j] = filename[i];
```

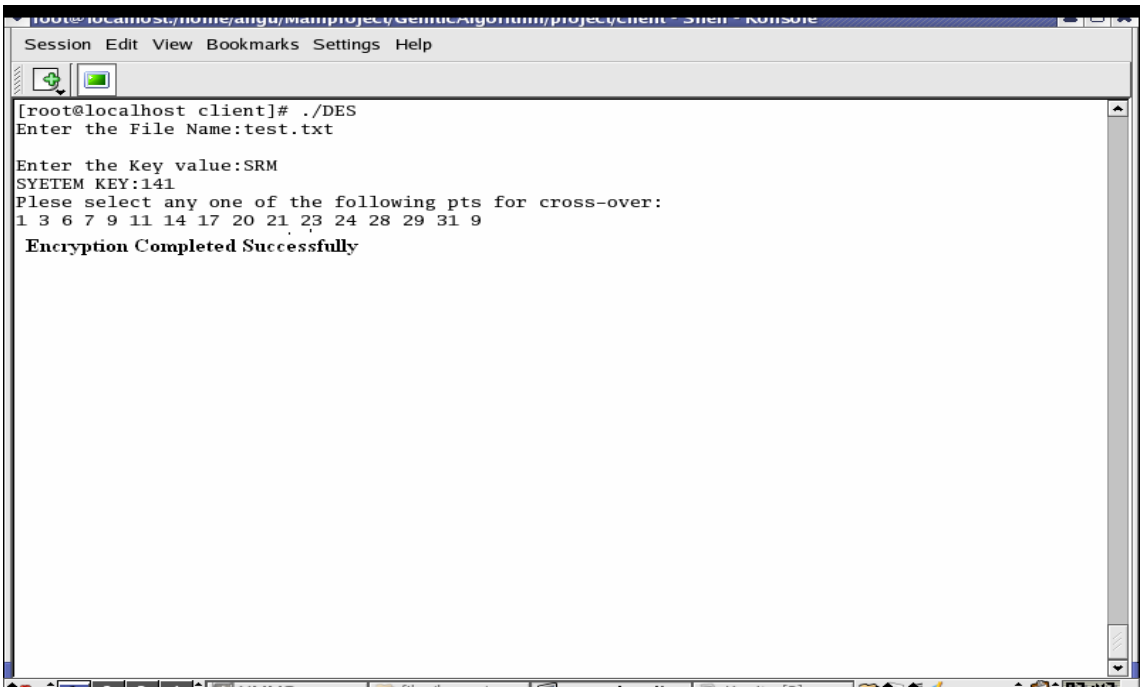
```
                j++;
        }
        strcat(temp_f,temp_ext);
}
/*********************************************************/
                /**********MDES.h**************/
```

APPENDIX-2

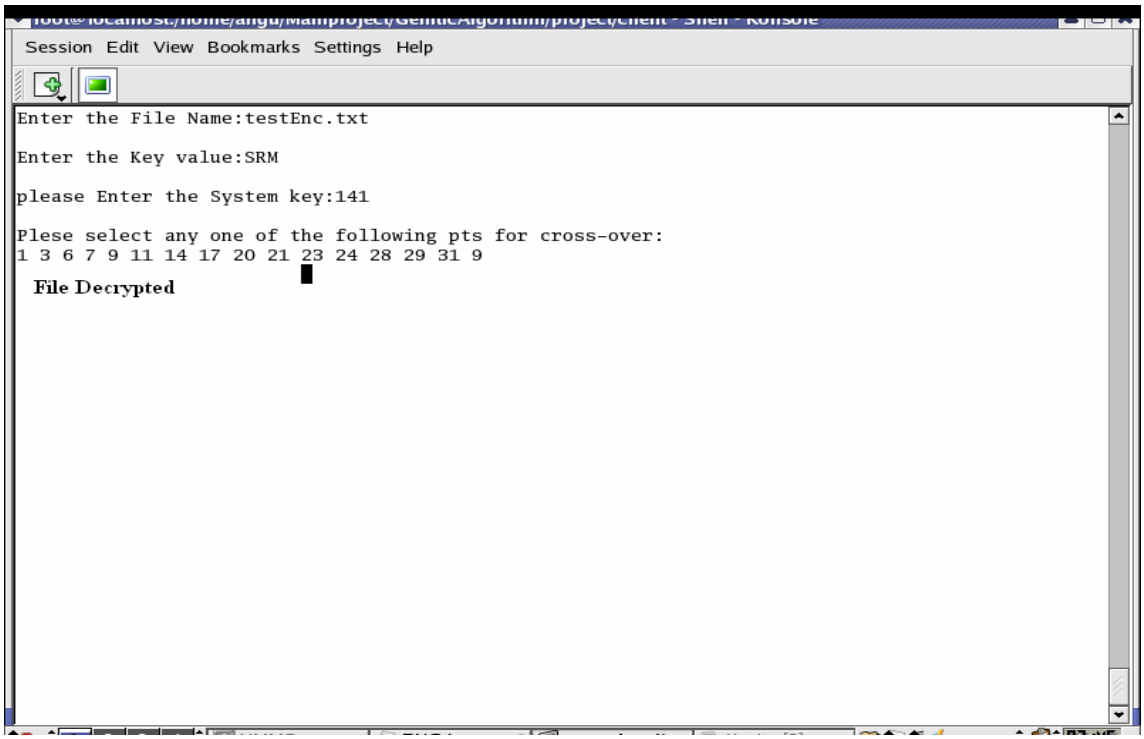
SAMPLE OUTPUTS

HOST LEVEL DETECTION:

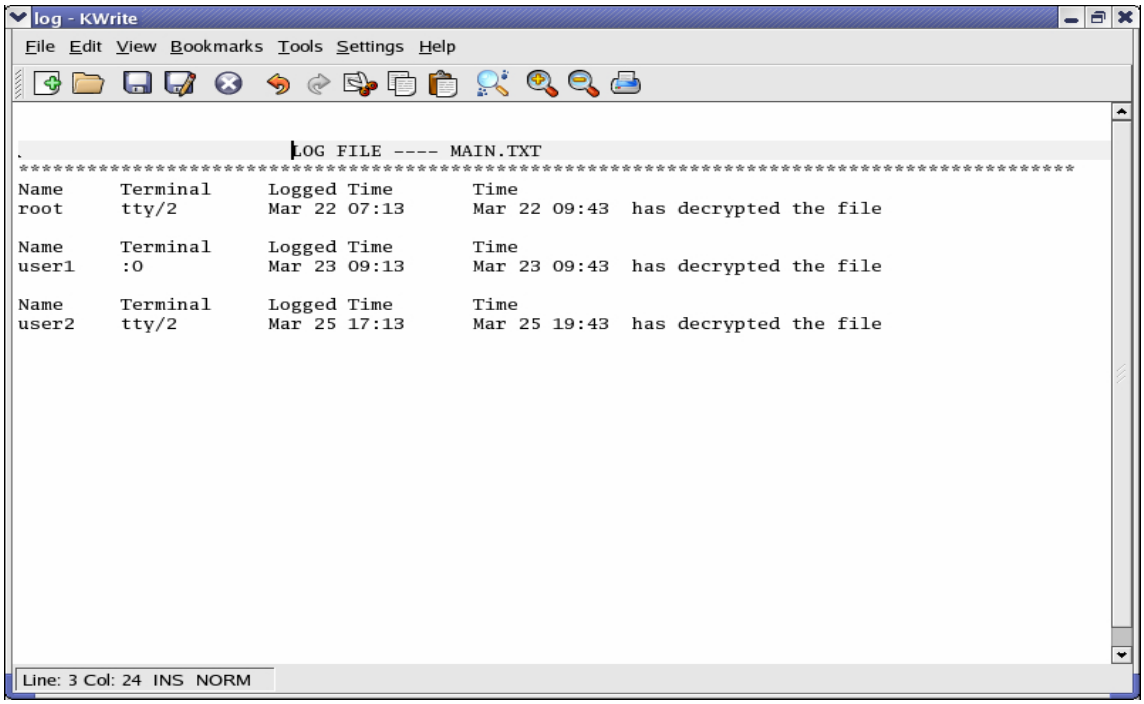
M-DES Encryption:



M-DES Decryption:

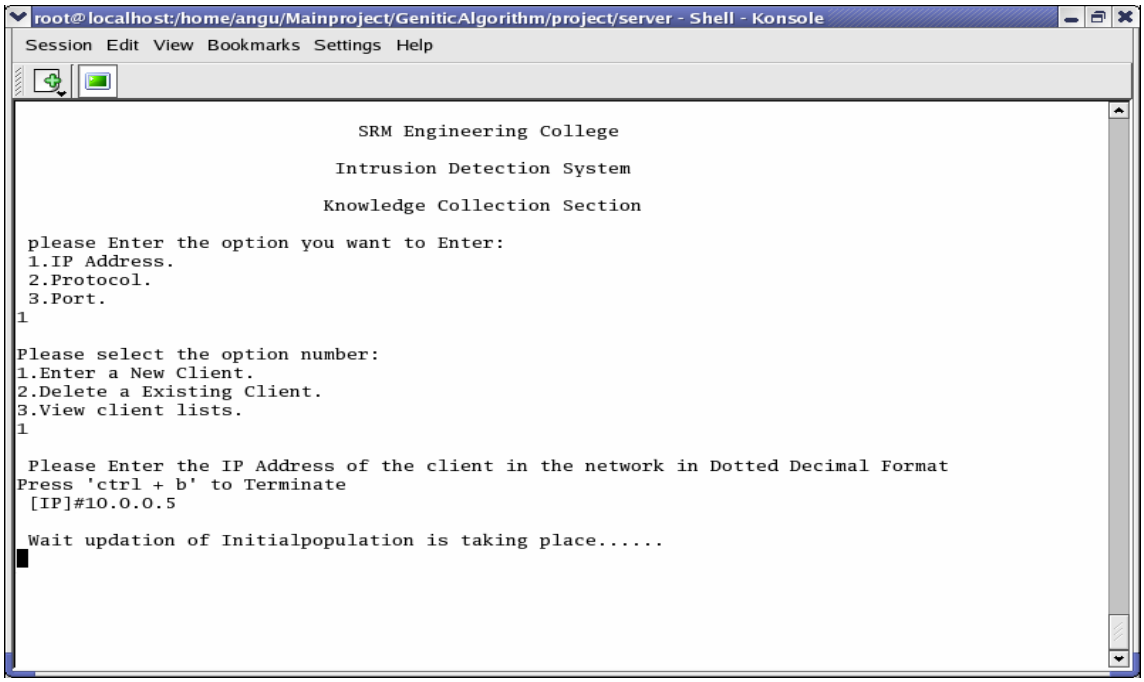


Los file for a File using MDES:

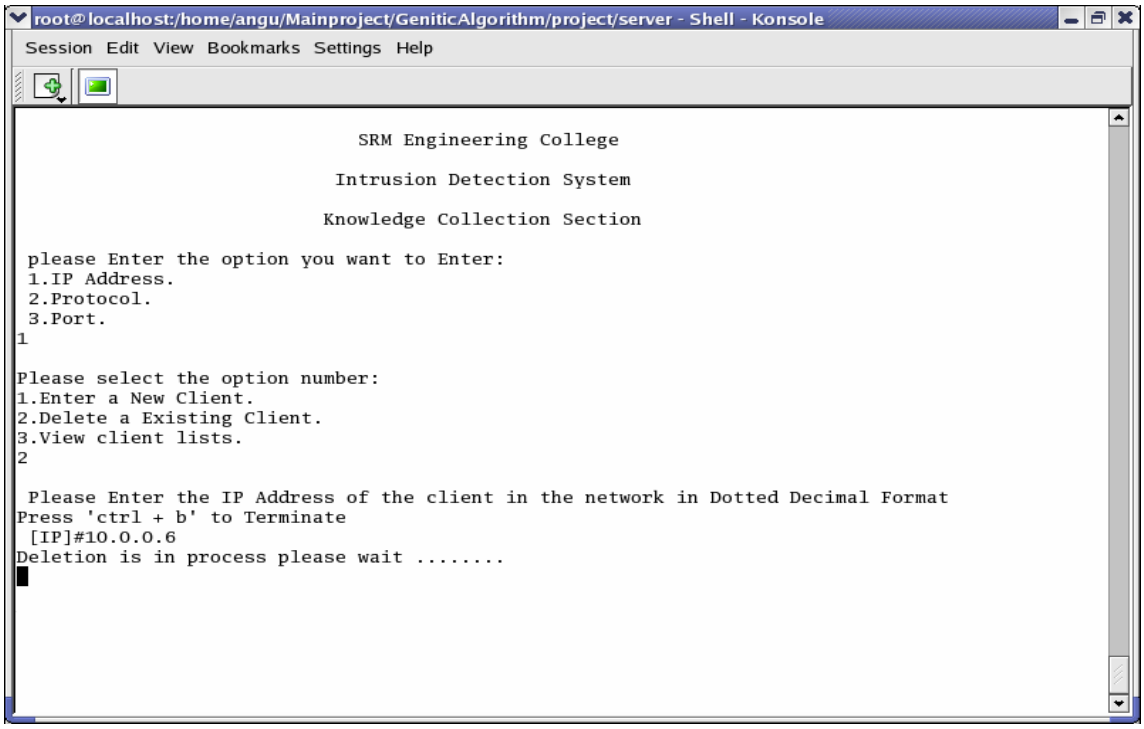


NETWORK LEVEL DETECTION:

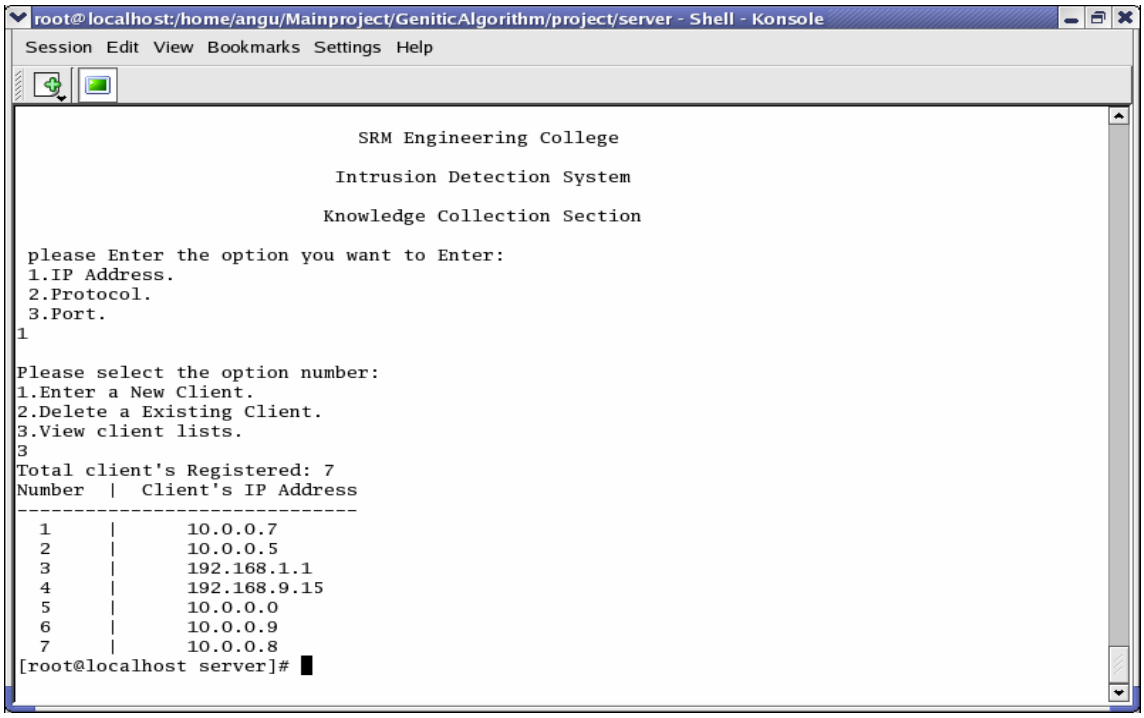
Knowledge Collection: Registration.



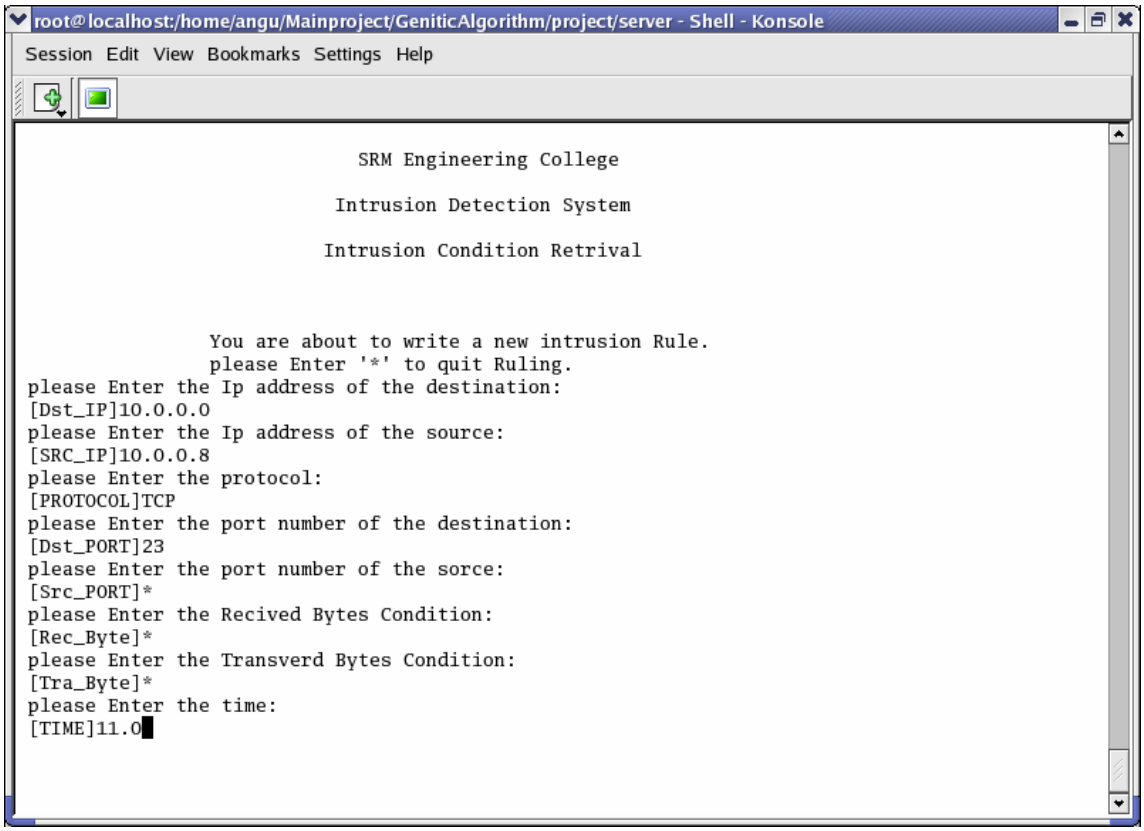
Knowledge Collection: Deletion



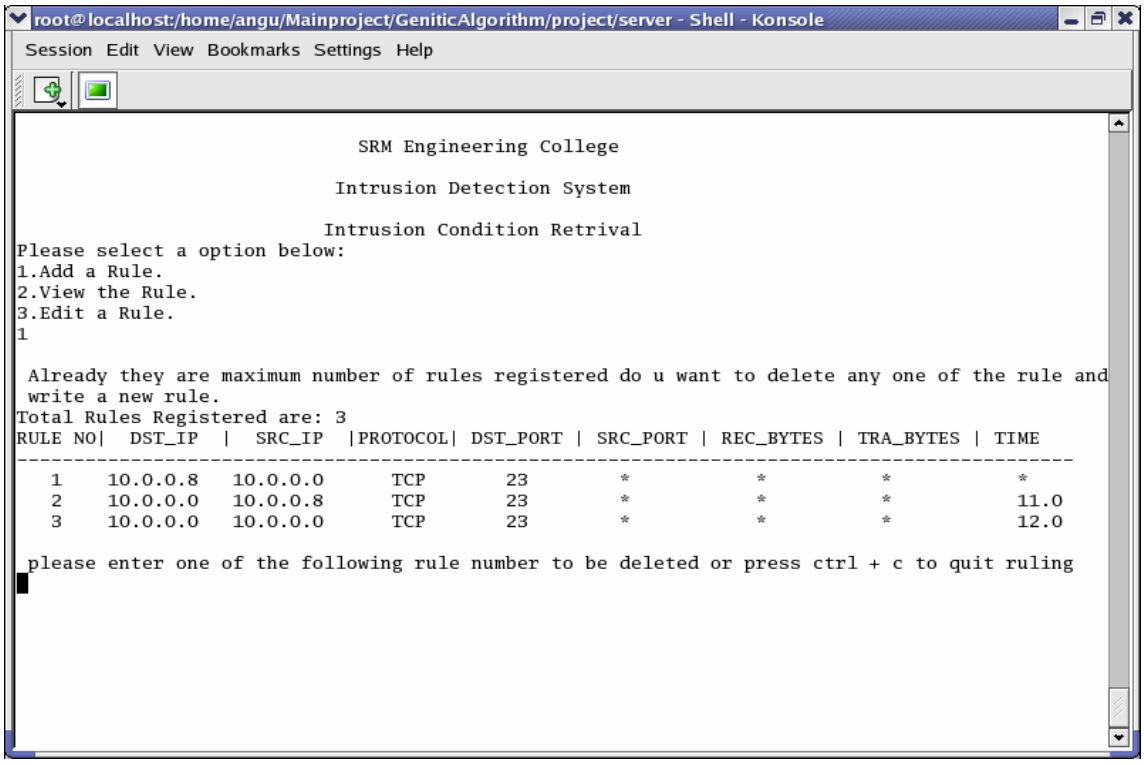
Knowledge Collection: View



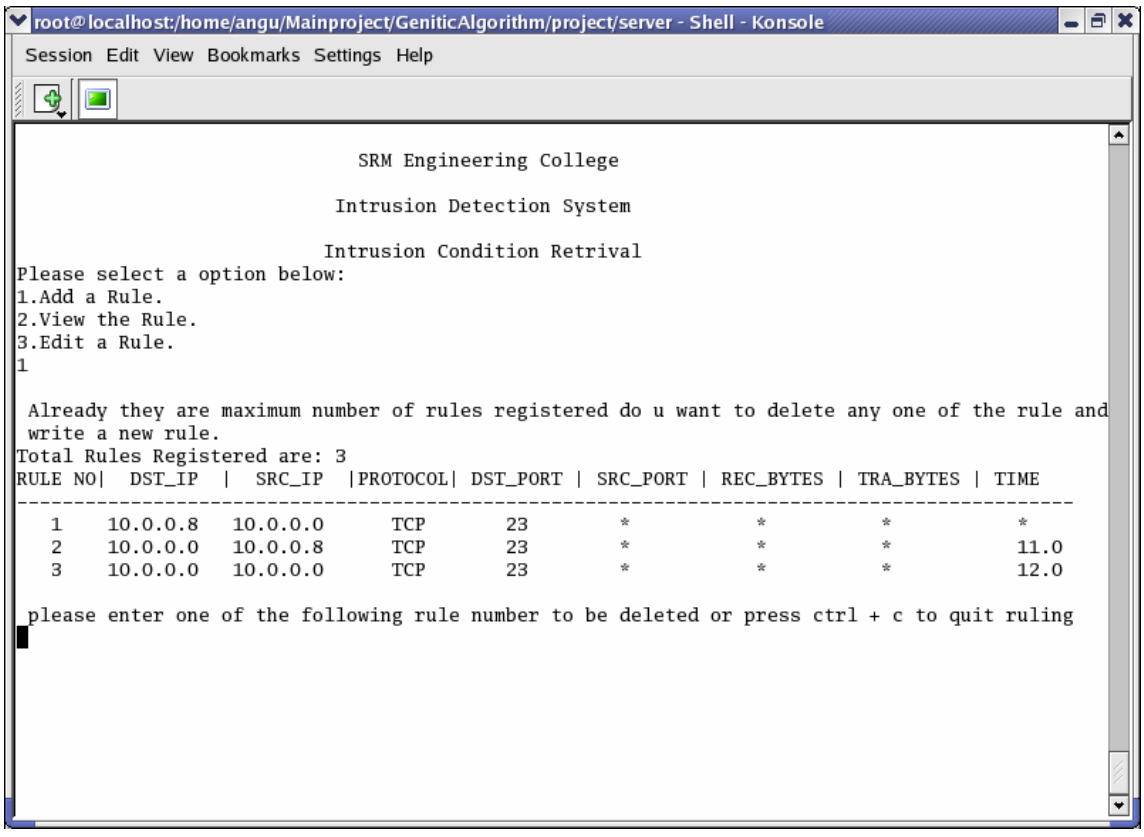
IDS Condition Retrieval:



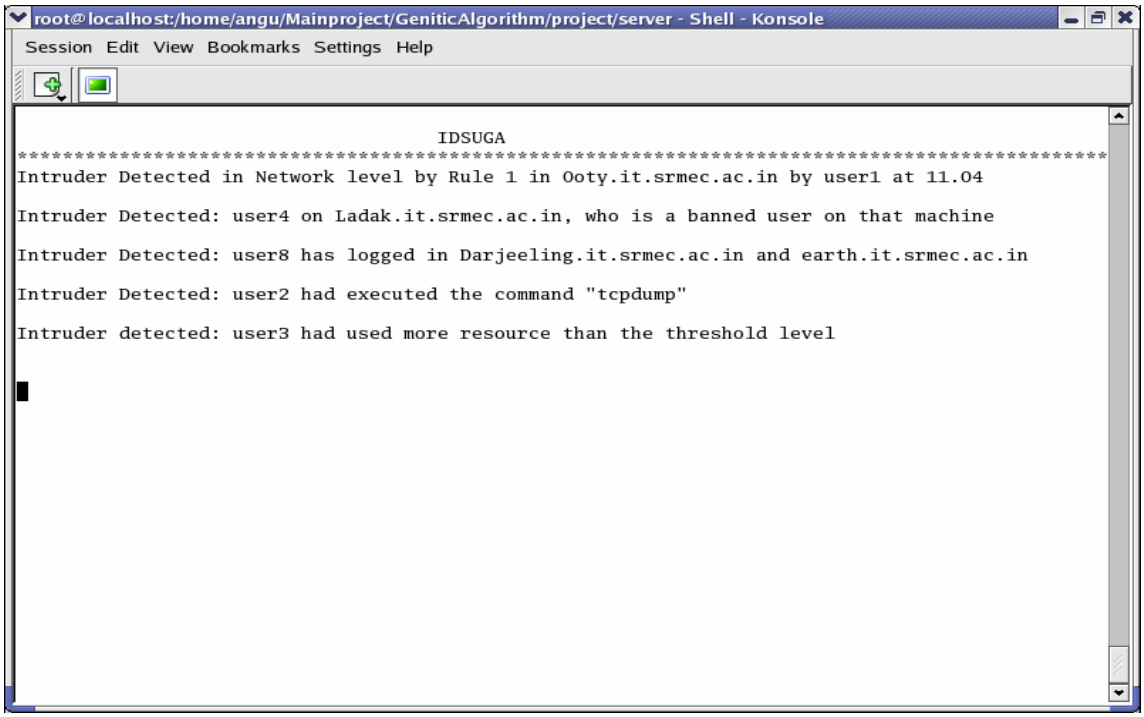
IDS Condition Deletion:



Rule sets Creation:



IDSUGA Server:


root@localhost:/home/angu/Mainproject/GeniticAlgorithm/project/server - Shell - Konsole
Session Edit View Bookmarks Settings Help
IDSUGA
Intruder Detected in Network level by Rule 1 in Ooty.it.srmec.ac.in by user1 at 11.04
Intruder Detected: user4 on Ladak.it.srmec.ac.in, who is a banned user on that machine
Intruder Detected: user8 has logged in Darjeeling.it.srmec.ac.in and earth.it.srmec.ac.in
Intruder Detected: user2 had executed the command "tcpdump"
Intruder detected: user3 had used more resource than the threshold level

REFERENCES

1) Bezroukov, Nikolai (30 Oct. 2003). "Intrusion Detection (general issues)." Softpanorama: Open Source Software Educational Society. URL: http://www.softpanorama.org/Security/intrusion_detection.shtml

2) Bridges, Susan, and Rayford B. Vaughn (2000). Intrusion Detection Via Fuzzy Data Mining. Proc. of 12th Annual Canadian Information Technology Security Symposium, Ottawa, Canada.

3) Crosbie, Mark, and Gene Spafford (1995). Applying Genetic Programming to Intrusion Detection. Proc. of 1995 AAAI Fall Symposium on Genetic Programming, Cambridge, Massachusetts.

URL: http://citeseer.nj.nec.com/crosbie95applying.html (30 Oct. 2003).

4) Goodberg (2001), "Genetic Algorithm: optimizing and searching" page 1-151

5) Jones, Anita. K. and Robert. S. Sielken.(2000) "Computer System Intrusion Detection: A Survey." Technical Report. Department of Computer Science, University of Virginia, Charlottesville, Virginia.

6) McHugh, John, (2001) "Intrusion and Intrusion Detection." Technical Report. CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University.

7) Paxson, Vern. Bro (1998): A System for Detecting Network Intruders in Real-time. Proc. of 7th USENIX Security Symposium, Jan. 26-29, San Antonio, Texas,: 31- 51.

8) Pohlheim, Hartmut.(2001) "Genetic and Evolutionary Algorithms: Principles, Methods and Algorithms." Genetic and Evolutionary Algorithm Toolbox.
URL: http://www.geatbx.com/docu/algindex.html.

9) Roesch, Martin (1999). Snort - Lightweight Intrusion Detection for Networks. Proc. Of 13th Systems Administration Conf. (LISA '99), Nov. 7-12, Seattle, Washington.
URL: www.snort.com

10) Sinclair, Chris, Lyn Pierce, and Sara Matzner.(2003) An Application of Machine Learning to Network Intrusion Detection. Proc. of 1999 Annual Computer Security Applications Conf. (ACSAC), Dec. 6-10, Phoenix, Arizona 1999: 371- 377.
URL: http://www.acsac.org/1999/papers/fri-b-030-sinclair.pdf

11) Whitley, Darrell. (1994) "A Genetic Algorithm Tutorial." Statistics and Computing 4 (1994): 65-85.
http://www.geatbx.com/docu/algindex.html (30 Oct. 2003).

12**)** Wei Li (2003), A Genetic Algorithm Approach to Network Intrusion Detection

13) Red Hat Linux Bible Version 10

14) Richards Stevenson, UNIX Network Programming. Volume 1,2.

15) Dougls.E.Comer "TCP/IP Illustrated" Volume 1.