CONtraCEPT
A Linux Based Firewall
Implemented On Knoppix O.S.

By

TEJAS BHATT-021202
MAYUR MANTRI-021402

विद्या तत्व ज्योतिसम:

JAYPEE UNIVERSITY OF
INFORMATION TECHNOLOGY

MAY-2006

Submitted in partial fulfillment of the degree of Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY-WAKNAGHAT

CERTIFICATE

This is to certify that the work entitled, "CONtraCEPT a linux based firewall implemented on knoppix O.S" submitted by Mr. Tejas Bhatt (On Roll :021202) and Mr. Mayur Mantri (On Roll :021402) in partial fulfillment for the award of Degree of Bachelor of Technology in CSE of Jaypee University of Information Technology (JUIT), Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

[illegible]

Lec. Chetna Dabbas
Project Coordinator

N. Prakash

Prof. Dr. Naveen Prakash
H.O.D Computer Science

ACKNOWLEDGEMENT

Many people have contributed to this project in a variety of ways over the past few months. We are sincerely thankful to Ms. Chetna Dabbas under whose supervision the whole project was carried out for her regular reviews and suggestions. We also acknowledge the many helpful comments received from other teachers of the different departments and visualization courses and seminars. We are deeply indebted to all those who provided reviews and suggestions for improving the materials and topics covered in our package, and we extend our apologies to anyone we may have failed to mention.

Thank You

TABLE OF CONTENTS

PROJECT SCHEDULE

SNo.	Activity Description	Planned Date	Actual Date	Status
1	Domain research.	7th March'06	12th March'06	Done
2	Learning about the inet deamon	10th March'06	12th March'06	Done
3	Requirement Analysis	15th March'06	15th March'06	Done
4	Modeling the concepts of the tool	20th March'06	20th March'06	Done
5	Designing the logical schema	22nd March'06	22nd March'06	Done
6	Designing the UI	24th March'06	25th March'06	Done
7	Mid Term Reporting	10th April'06	10th April '06	Done
8	Implementation of different modules	15th April'06	19th April'06	Done
9	Integration of the modules	2nd May'06	2nd May'06	Done
10	Testing	8th May'06	10th May'06	Done
11	Reporting	11th May'06	11th May'06	Done

LIST OF FIGURES

LIST of ABBREVIATIONS

BSD	Berkeley Software Distribution
BPD	Border Protection Device
DOS	Denial of Service
GUI	Graphical User Interface
IP	Internet Protocol
MAC	Media Access Control
OS	Operating System
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WAN	Wide Area Network

ABSTRACT

The erstwhile question of security continues to pose challenges to OS developers and programmers in general around the world. It has become increasingly important for any individual who owns a computer connected to any kind of network to have a well defined security policy as a defense mechanism against any kind of intrusion attempt.A firewall, also called a **Border Protection Device** (**BPD**), especially in NATO contexts, or **packet filter** in BSD contexts thus has become an integral part of the suite of software installed on a computer. The project is an endeavor in this direction in which a software firewall (CONtraCEPT) is built for a network enabled linux/debian based system.

CHAPTER 1: A FIREWALL PRIMER

In computing, a **firewall** is a piece of hardware and/or software which functions in a networked environment to prevent some communications forbidden by the security policy, analogous to the function of firewalls in building construction.

A firewall has the basic task of controlling traffic between different zones of trust. Typical zones of trust include the Internet (a zone with no trust) and an internal network (a zone with high trust). The ultimate goal is to provide controlled connectivity between zones of differing trust levels through the enforcement of a security policy and connectivity model based on the least privilege principle.

Least privilege principle

In computer science and other fields the **principle of minimal privilege**, also known as **principle of least privilege** or just **least privilege**, requires that in a particular abstraction layer of a computing environment every module (which can be for example, a process, a user or a program on the basis of the layer we are considering) must be able to see only such information and resources that are immediately necessary.

So the idea of the principle is to grant just the minimum possible privileges to permit a legitimate action, in order to enhance protection of data and functionality from faults (fault tolerance) and malicious behaviour (computer security).

Types of firewalls

There are three basic types of firewalls depending on:

- Whether the communication is being done between a single node and the network, or between two or more networks.
- Whether the communication is intercepted at the network layer, or at the application layer.
- Whether the communication state is being tracked at the firewall or not.

With regard to the scope of filtered communications there exist:

- Personal firewalls, a software application which normally filters traffic entering or leaving a single computer.
- Network firewalls, normally running on a dedicated network device or computer positioned on the boundary of two or more networks or DMZs (demilitarized zones). Such a firewall filters all traffic entering or leaving the connected networks.

The latter definition corresponds to the conventional, traditional meaning of "firewall" in networking.

In reference to the layers where the traffic can be intercepted, three main categories of firewalls exist:

- Network layer firewalls. An example would be iptables.
- Application layer firewalls. An example would be TCP Wrapper.
- Application firewalls. An example would be restricting ftp services through /etc/ftpaccess file

These network-layer and application-layer types of firewall may overlap, even though the personal firewall does not serve a network; indeed, single systems have implemented both together.

There's also the notion of application firewalls which are sometimes used during wide area network (WAN) networking on the world-wide web and govern the system software. An extended description would place them lower than application layer firewalls, indeed at the Operating System layer, and could alternately be called operating system firewalls. Some firewalls have higher privileges than others like mysql and pj.

Lastly, depending on whether the firewalls track packet states, two additional categories of firewalls exist:

- Stateful firewalls
- Stateless firewalls

Network layer firewalls

In computer networks, a **network layer firewall** works as a packet filter by deciding what packets will pass the firewall according to rules defined by the administrator. Filtering rules can act on the basis of source and destination address and on ports, in addition to whatever higher-level network protocols the packet contains. Network layer firewalls tend to operate very fast, and transparently to users.

Network layer firewalls generally fall into two sub-categories, stateful and non-stateful. Stateful firewalls hold some information on the state of connections (for example: established or not, initiation, handshaking, data or breaking down the connection) as part of their rules (e.g. only hosts inside the firewall can establish connections on a certain port).

Stateless firewalls have packet-filtering capabilities but cannot make more complex decisions on what stage communications between hosts have reached. Stateless firewalls therefore offer less security. Stateless firewalls somewhat resemble a router in their ability to filter packets.

Any normal computer running an operating system which supports packet filtering and routing can function as a network layer firewall. Appropriate operating systems for such a configuration include Linux, Solaris, BSDs or Windows Server.

CONtraCEPT is a stateless, network layer firewall.

CHAPTER 2: A BRIEF INTRODUCTION

Using the inetd Daemon

Each server running under UNIX offering a service normally executes as a separate process. When the number of services being offered becomes large, however, this becomes a burden to the system.This is because resources must be allocated to each server process running, even when there are no current requests for the services being offered.
Additionally, it can be observed that most server programs use the same general procedure to create, bind, listen, and accept new client connections. A similar observation can be made for connectionless server operation.

Steps Common to Most Servers

The basic steps a connection-oriented server uses to establish contact with a client were the following:
1. Create a socket.
2. Bind a socket to a well-known address.
3. Listen for a client connect.
4. Accept the client connect.

You will see that the inetd daemon can perform these initial steps for any connection-oriented server, saving the server writer from having to write and debug code for these steps. The inetd daemon idea can be extended to handle connectionless servers as well.

Introducing inetd

When a Linux system is booted for the first time, the inetd daemon is started from one of the startup scripts. On Knoppix 3.8 systems, this daemon is started from the script file:

/usr/sbin/inetd

When the inetd daemon is started for the first time, it must know what Internet services it must listen for and what servers to pass the request off to when a request arrives. This is defined within the startup file /etc/inetd.conf.

The /etc/inetd.conf Configuration File

The general file layout of the /etc/inetd.conf file is organized as a text file, with each text line representing one record, which describes one Internet service. Lines starting with # are simply comment lines and are ignored.
The blank (or tab) separated fields are described in Table with some examples (fields are listed in order from left to right).

Field # Description Example

1. Internet service name telnet (this might also be a port number)
2. Socket type stream or dgram
3. Protocol tcp or udp
4. Flags nowait or wait
5. Userid to use root or nobody
6. Pathname of executable /usr/sbin/in.telnetd
7. Server arguments in.telnetd

The Design Parameters of inetd Servers

One of the advantages of using inetd as the front end for servers is that the server writer's job is made easier. There is no longer the burden of writing the same Socket(2), bind(2), listen(2), and accept(2) calls for stream tcp servers, for example.
Similar code savings can be had for dgram udp servers, also. How then, does the inetd server hand off the connected socket to the server process when the process is started? Using the simple elegance of UNIX, the started server is handed the client socket on the following file units (file descriptors):

- File unit 0 has client socket for standard input
- File unit 1 has client socket for standard output
- File unit 2 has client socket for standard error

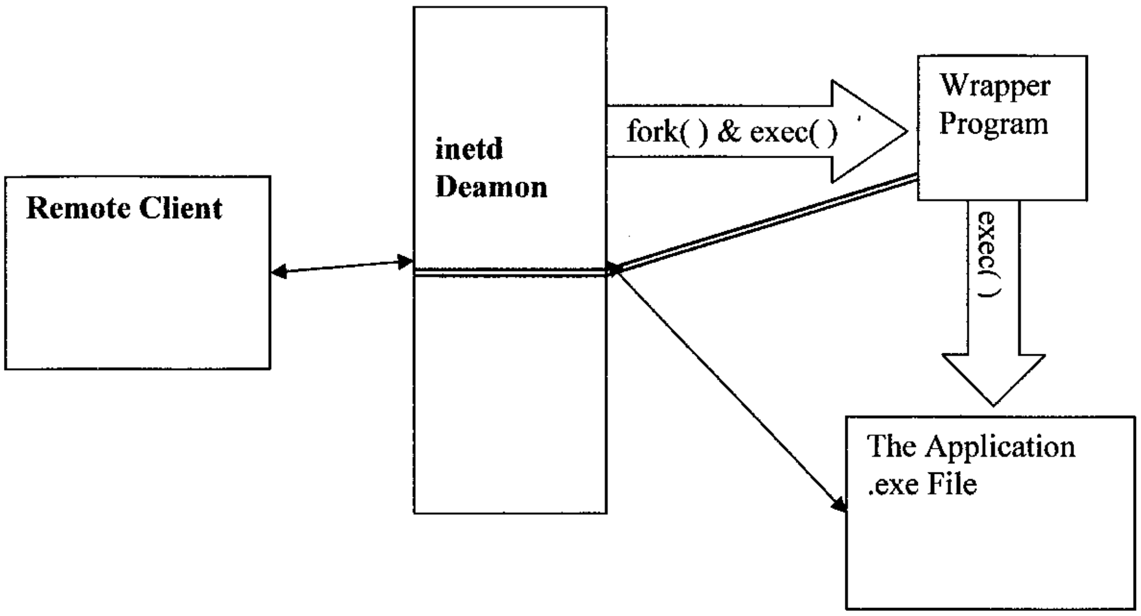


Figure 1: Graphical Representation of TCP wrapper concept

CHAPTER 3: DESIGN SPECIFICATIONS

Context Diagram

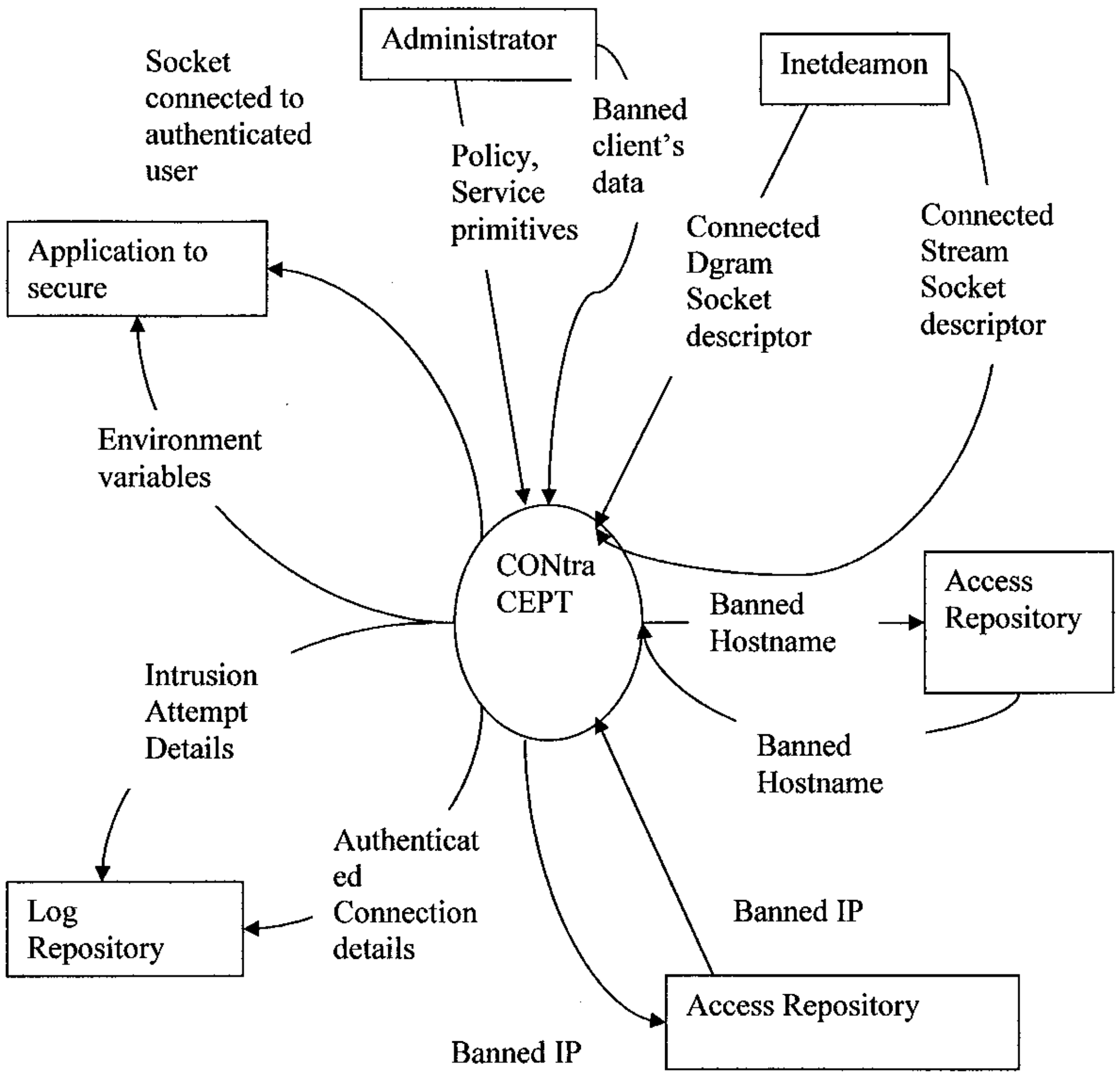


Figure 2: Context Diagram

Event List

Sno.	Event Name	Stimulus (Input)	Response (Output)	Type
1	Arrival of a connection request	Connected Socket descriptor on standard input	The authentication status message, socket connected to authenticated client, log entry corresponding to the authentication status	Flow
2	Arrival of a request to Secure a new service/application	Security Policy and service primitives	The new service is added to the list of services to be secured	Flow
3	Arrival of a request to terminate the firewall	Request to terminate firewall	The firewall is terminated	Flow
4	Arrival of request to populate access list	Request to add banned ip or banned hostname	The banned hostname or ip is added to the access list	Flow

Cartesian Hierarchy

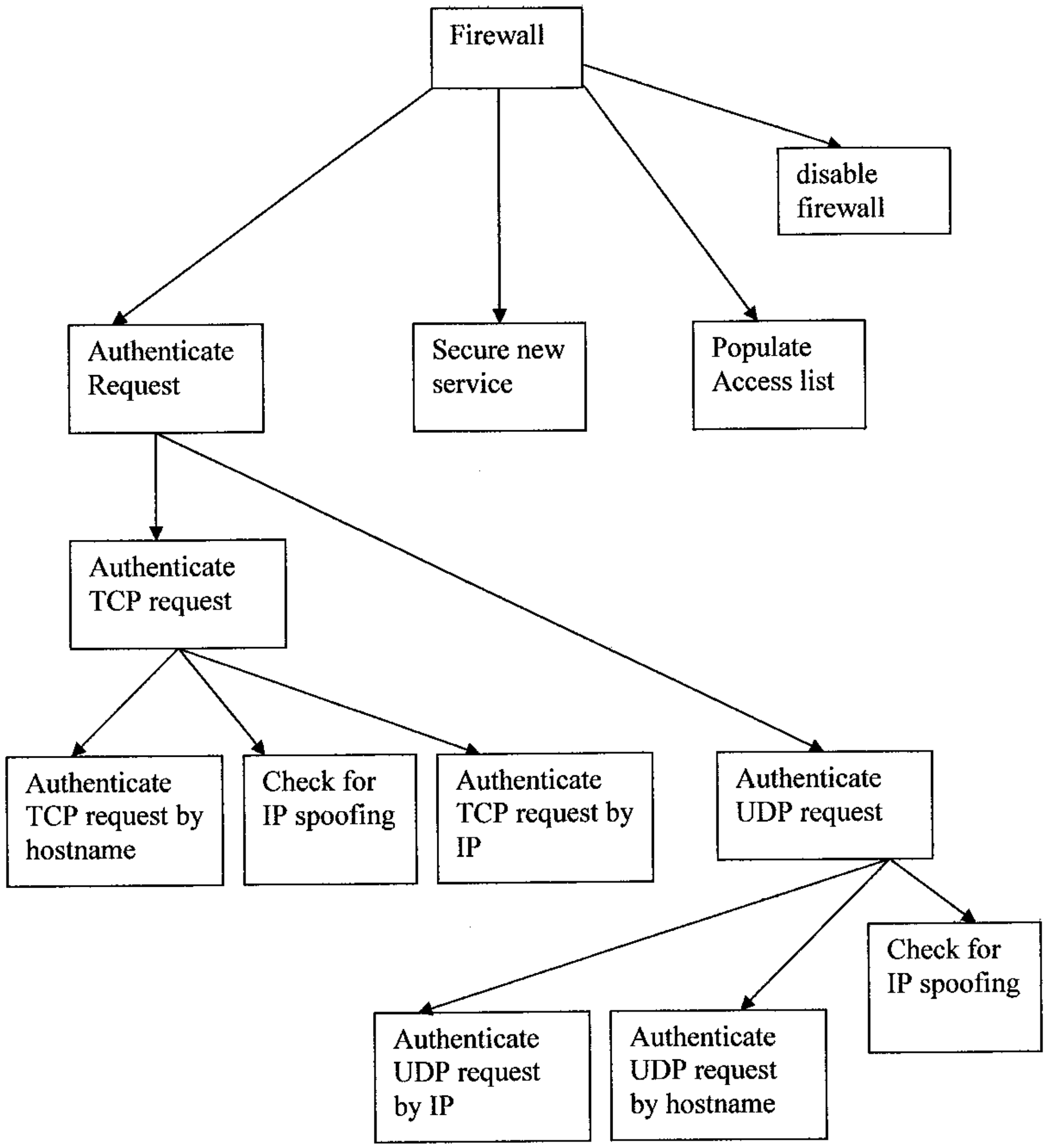


Figure 3: Cartesian Hierarchy

Data Flow Diagram

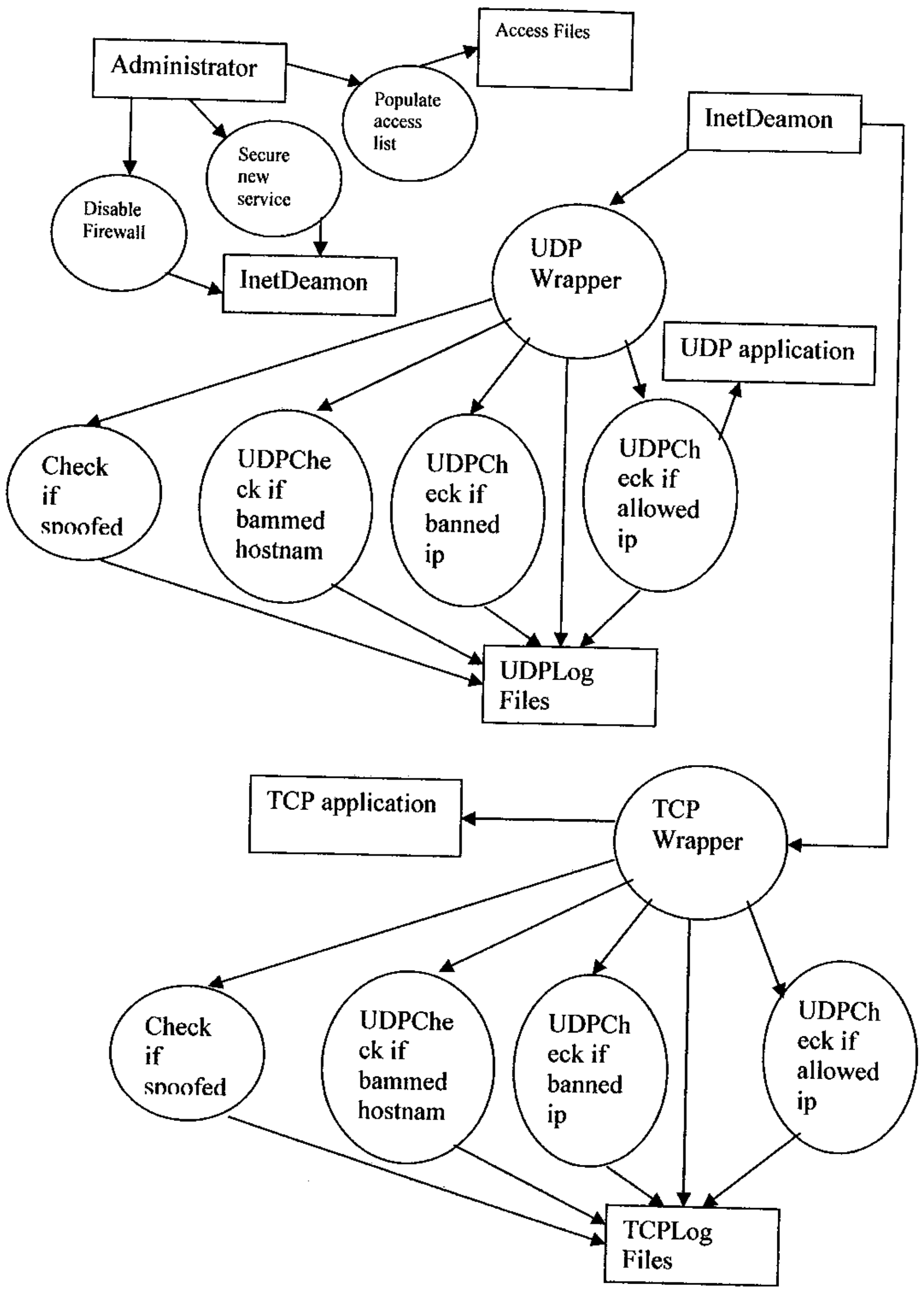


Figure 4: Data Flow Diagram

CHAPTER 4: IMPLEMENTATION SPECIFICATION

<u>The .c files of the project along with their key functions</u>

TCPWrapper.c

- Responsibility:
  This c file is responsible for the authentication and logging of any TCP connection request.

- Arguments:
  It accepts the full application path , policy variable and environment variables(implicit) as arguments.

Key functions

Sno.	**Function Name**	**(Input)**	**Response (Output)**
1	CheckifBannedip()	Pointers to a hostent structure and sockaddr_in structure , both filled with clients address information	Log entry corresponding to the authentication status of the client
2	Checkifbannedhostname()	Pointers to a hostent structure and sockaddr_in structure , both filled with clients address information	Log entry corresponding to the authentication status of the client

Sno.	**Function Name**	**(Input)**	**Response (Output)**
3	Checkifspoofed()	Pointers to a hostent structure and sockaddr_in structure , both filled with clients address information	Log entry corresponding to the authentication status of the client
4	Checkifallowedip()	Pointers to a hostent structure and sockaddr_in structure , both filled with clients address information	Log entry corresponding to the authentication status of the client

UDPWrapper.c

- Responsibility:
  This c file is responsible for the authentication and logging of any UDP connection request.

- Arguments:
  It accepts the full application path , policy variable and environment variables(implicit) as arguments.

Key functions

Sno.	**Function Name**	**(Input)**	**Response (Output)**
1	CheckifBannedip()	Pointers to a hostent structure and sockaddr_in structure , both filled with clients address information	Log entry corresponding to the authentication status of the client
2	Checkifbannedhostname()	Pointers to a hostent structure and sockaddr_in structure , both filled with clients address information	Log entry corresponding to the authentication status of the client
3	Checkifspoofed()	Pointers to a hostent structure and sockaddr_in structure , both filled with clients address information	Log entry corresponding to the authentication status of the client

Sno.	Function Name	(Input)	Response (Output)
4	Checkifallowedip()	Pointers to a hostent structure and sockaddr_in structure , both filled with clients address information	Log entry corresponding to the authentication status of the client

Log.c:

- Responsibilities:
  This non-executable c file holds all the functions responsible for the logging of information into the log files.

Sno.	**Function Name**	**(Input)**	**Response (Output)**
1	Log_open()	Pathneme of file to open, and mode in which it is to be opened.	File pointer to the file opened
2	Log1()	Message to be logged into the logfile and the format specifier	Message is logged into the corresponding log file
3	Log_close()	File pointer to the file to be closed	The file is closed and the memory associated with the file pointer is freed.
4	Bail()	On what, errorcode	Error message is logged is logged.

gadmin.c

- Responsibility:
  To show the CUI at the server side.

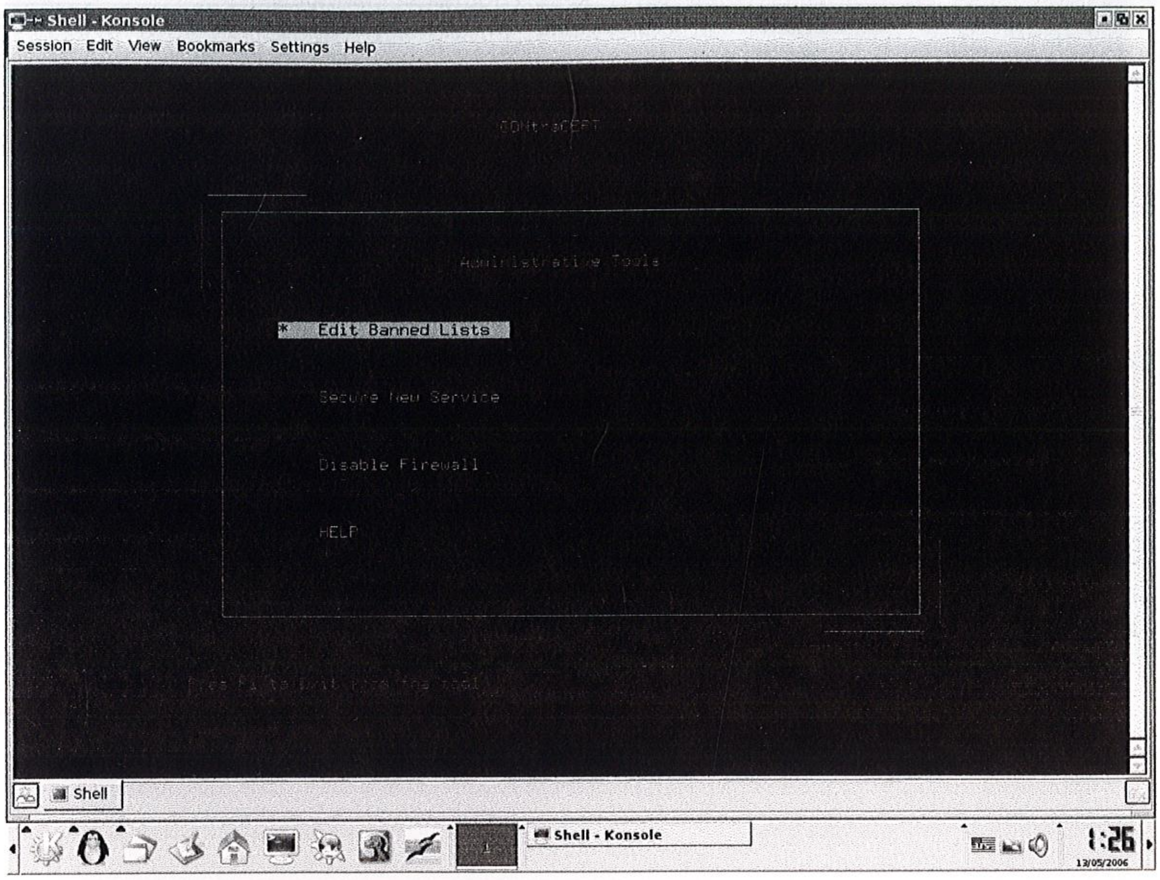
Key functions

Sno.	**Event Name**	**(Input)**	**Response (Output)**
1	appendEntryConf()	void	Append entries into the ined.conf file for starting of services.
2	appendEntryLog()	Index which specifies the log file to be edited	Any of the log files viz. Bannedip.log Allowedip.log Bannedalias.log edited.
3	adminMainMenu()	void	Return the position of the menu item selected by the administrator.
4	editLogFile()	void	Returns the target index to the log file.
5.	helpPage()	void	Displays the help file

gclient.c

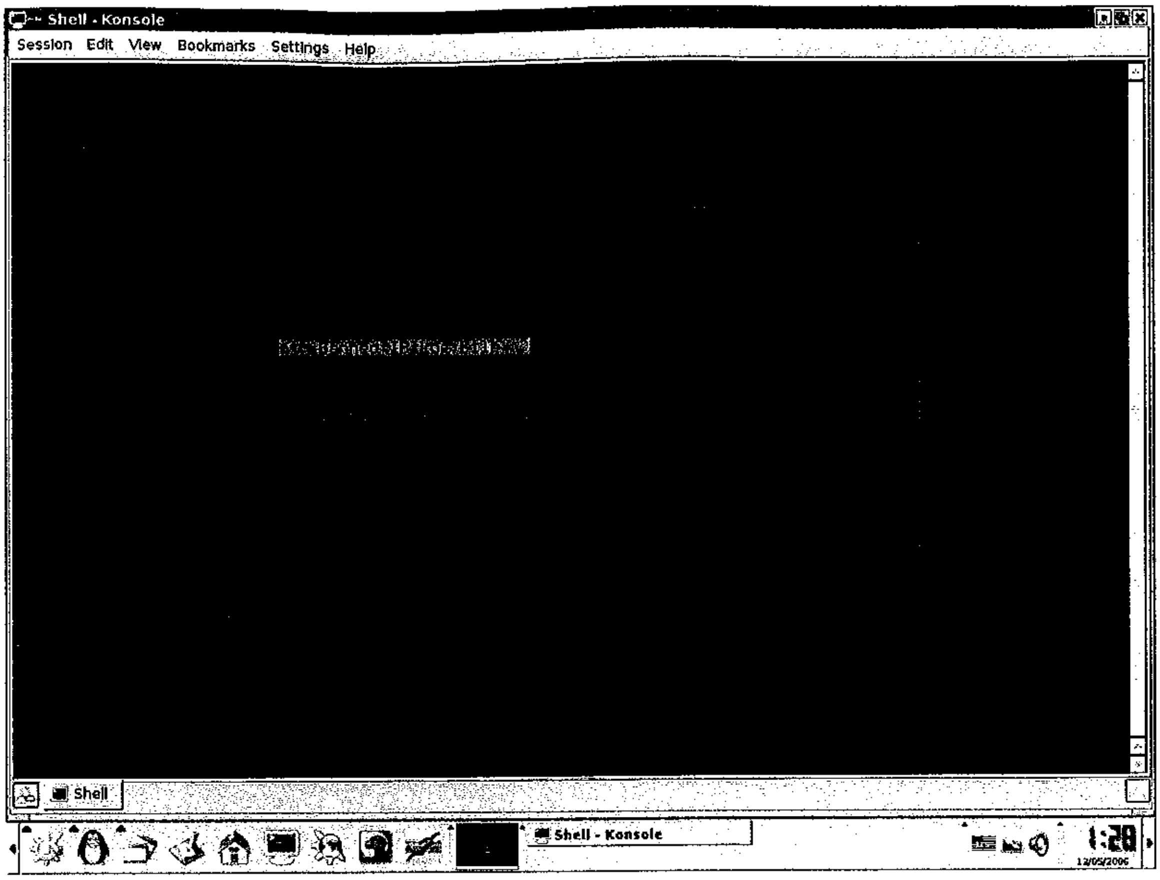
- Responsibility:
  To show the CUI at the client side.

CHPTER 5: SNAPSHOTS OF THE TOOL



```
                              CONtraCERT

                         Administrative Tools

      *   Edit Banned Lists

          Secure New Service

          Disable Firewall

          HELP

     Press F1 to Exit from the tool
```

Figure 5: Snapshot of Main Menu



```
[illegible]
```

Figure 6: Snapshot of Administrative Tools→Edit Banned Lists

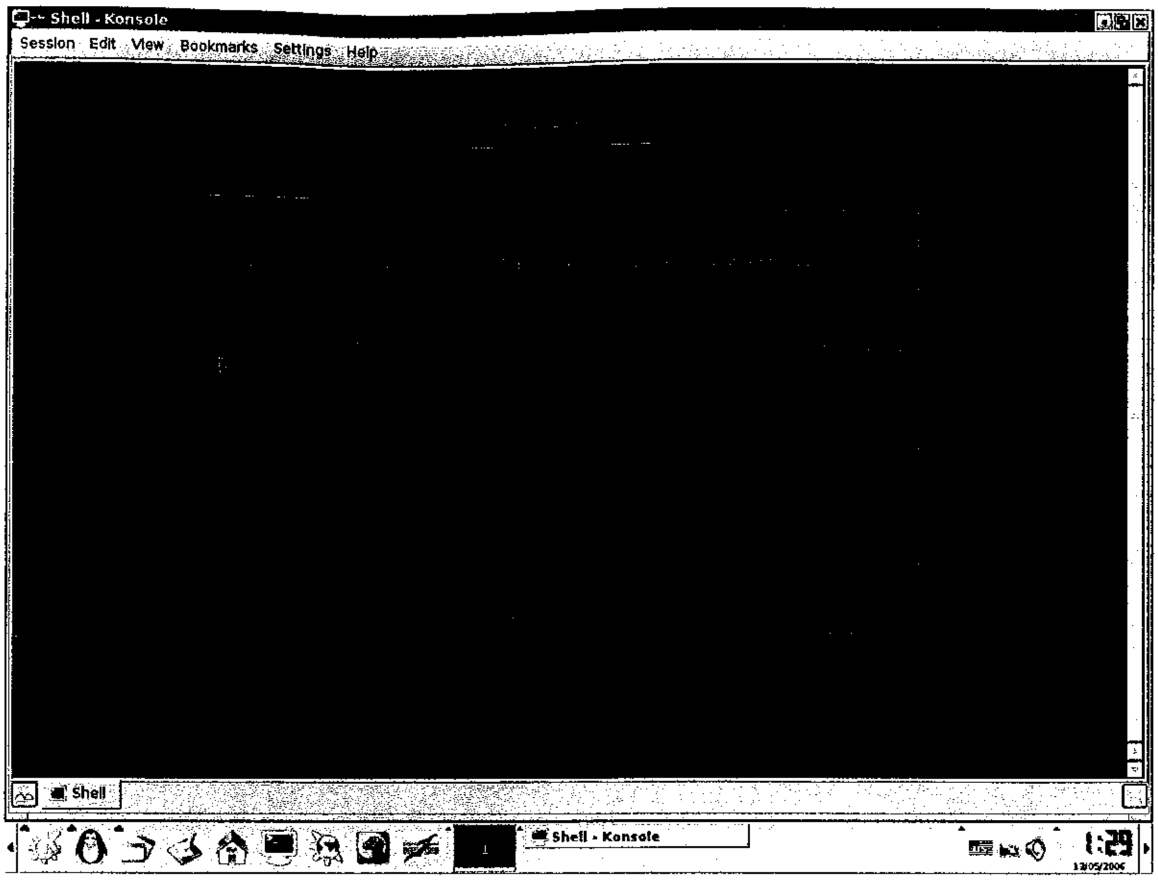


Figure 7: Snapshot of Banned IP log File →bannedip.log

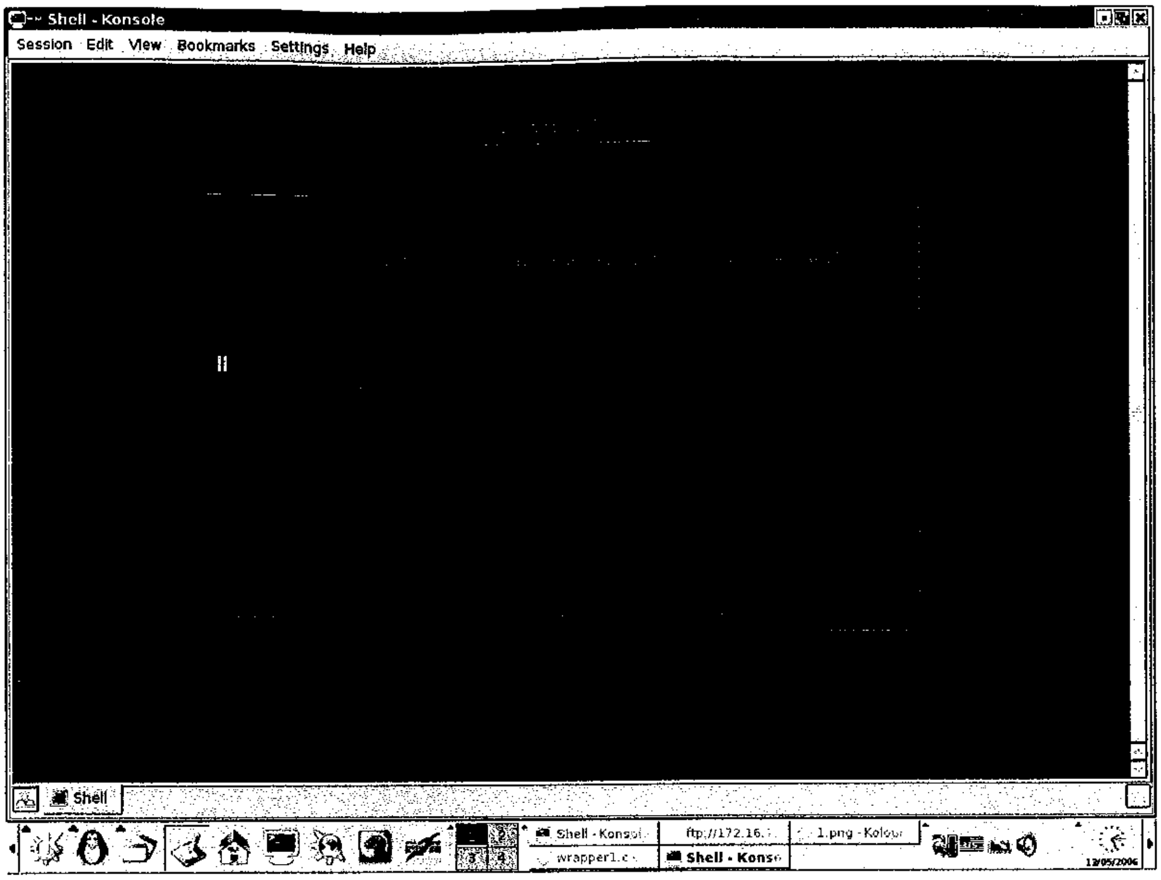


Figure 8: Snapshot of Administrative Tools→Edit Banned Lists→bannedalias.log

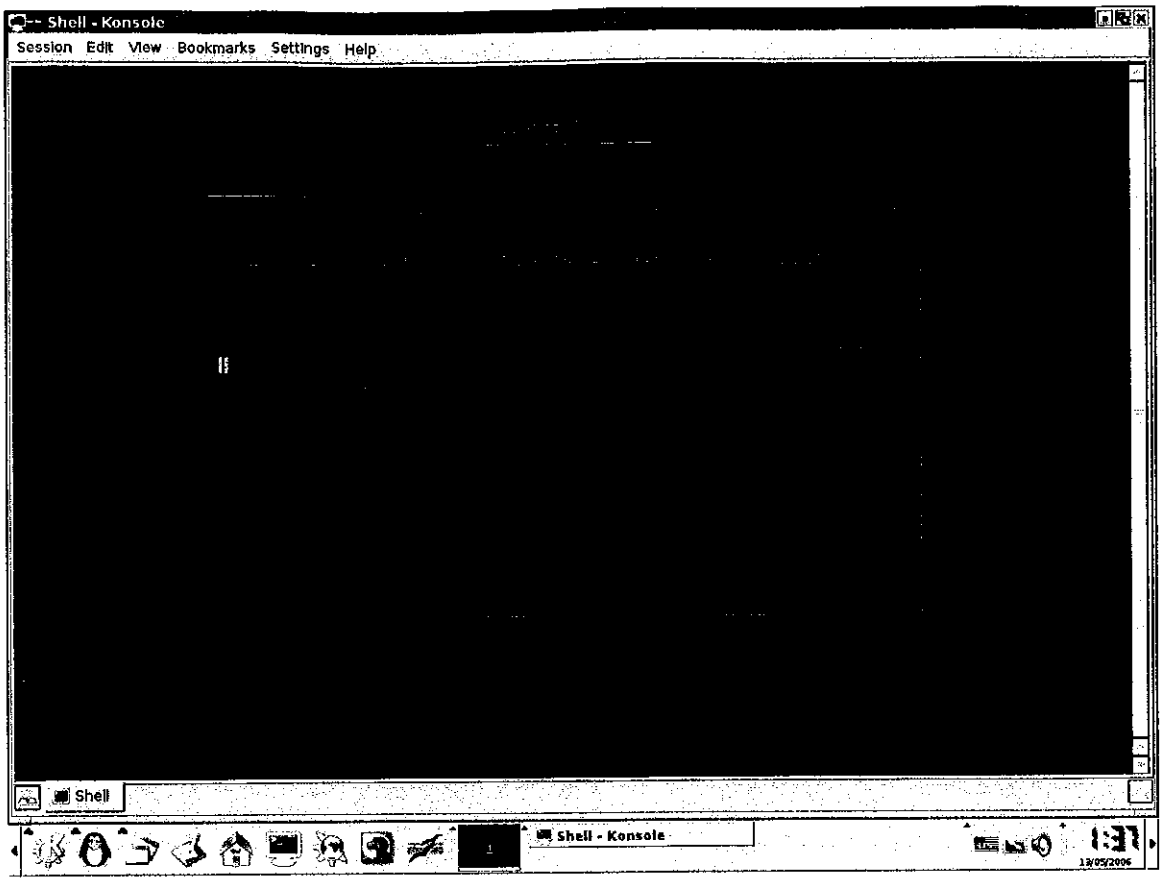


Figure 9: Snapshot of Administrative Tools→Edit Banned Lists→allowedip.log

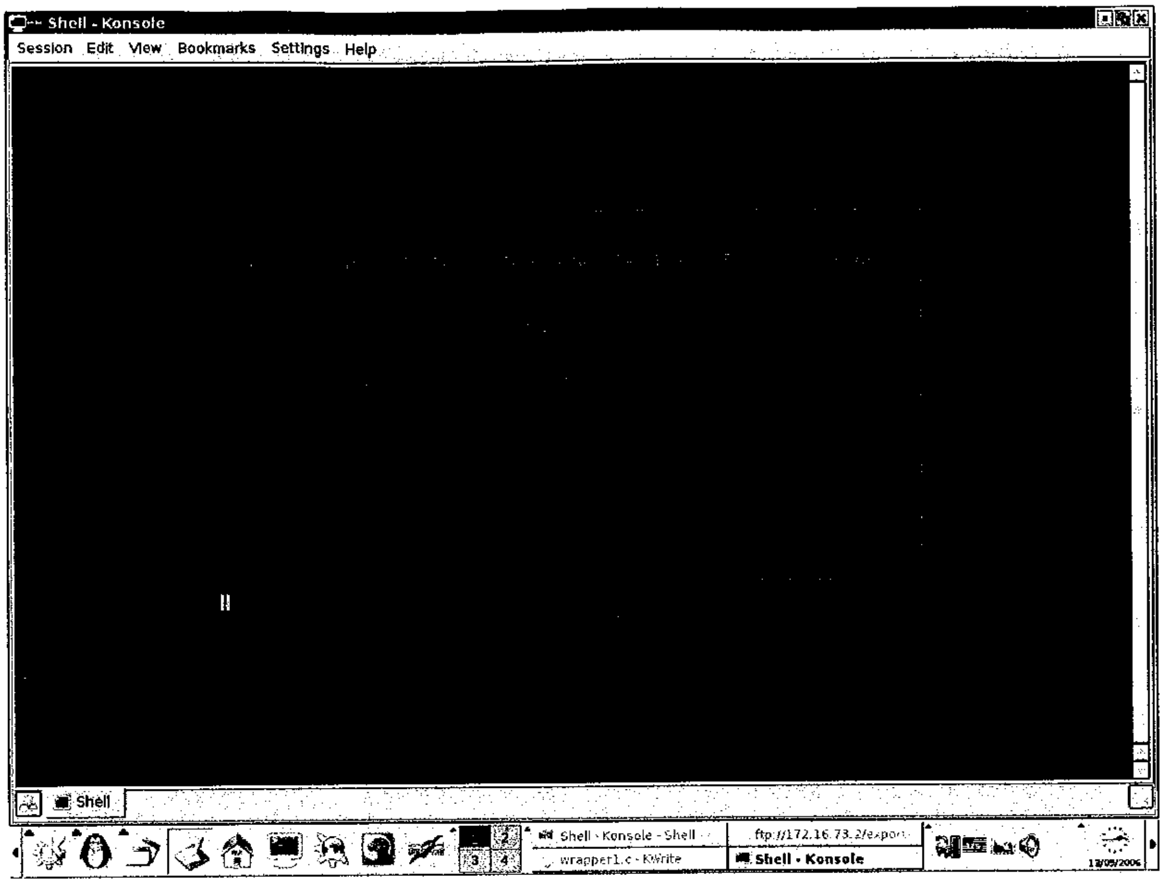


Figure 10: Snapshot of Administrative Tools→Secure New Service →Edit inetd.conf

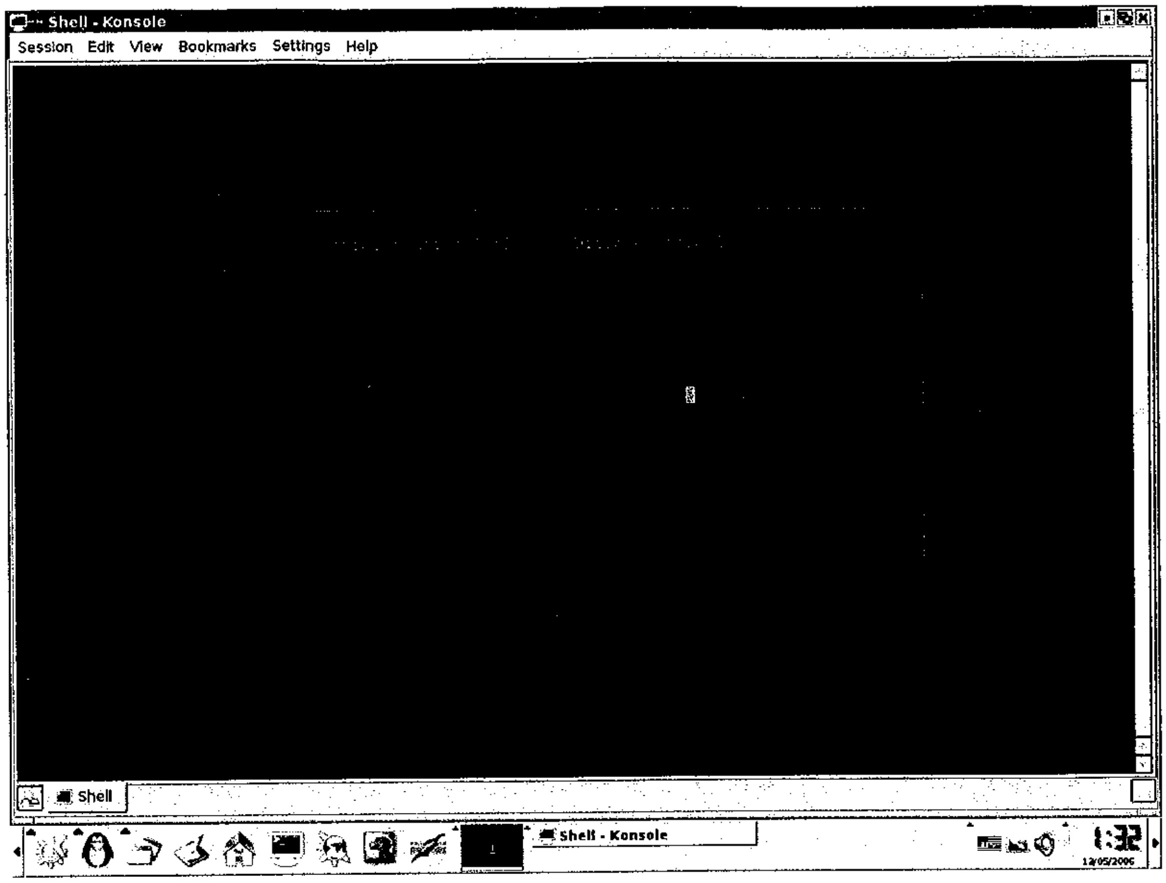


Figure 11: Snapshot of Administrative Tools→Disable Firewall

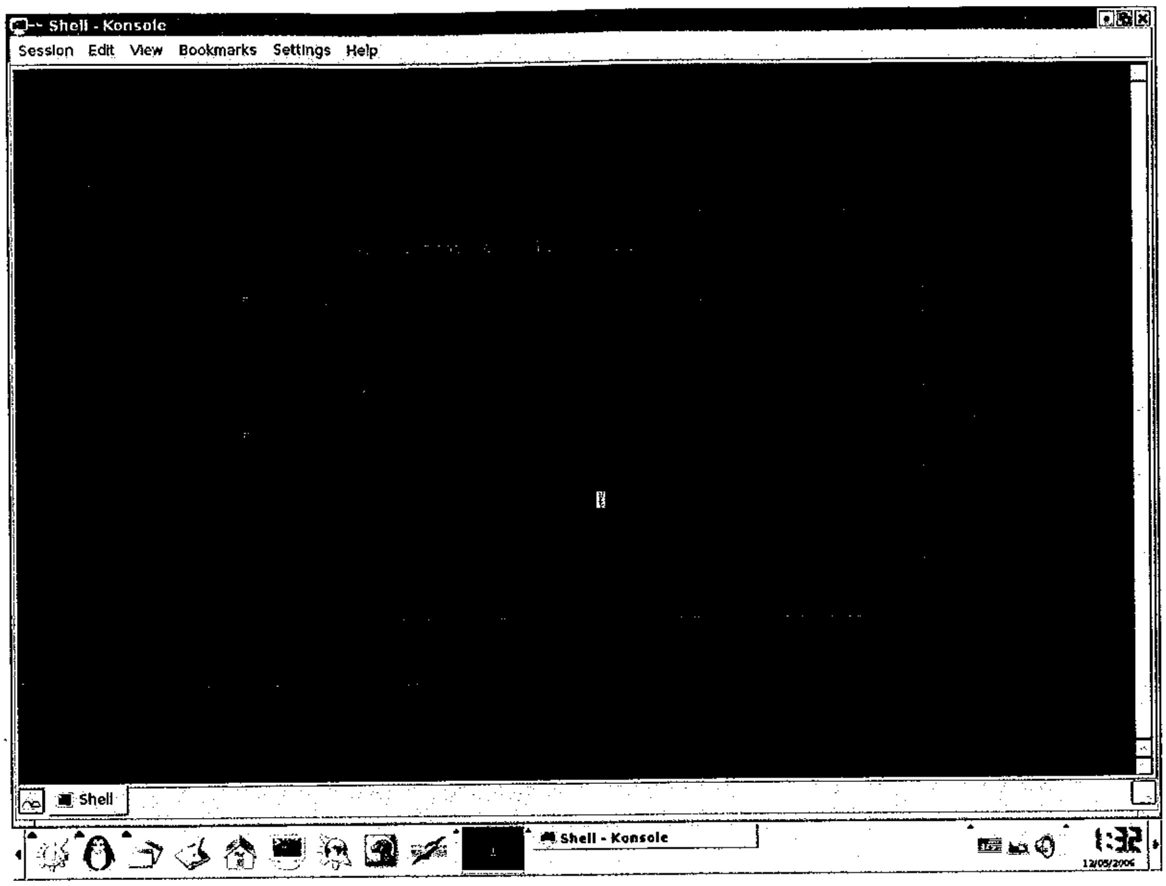


Figure 12: Snapshot of Administrative Tools→HELP

CHAPTER 6: LIMITATIONS

Major limitations of the tool are :

1. No prevention from attacks like DOS attacks.
2. IP spoofing by changing MAC address is not detected by the tool.
3. CUI does not allow allow all administrative options to be handled from it only.
4. *Virulent Data Patterns* cannot be recognized once the connection is made.

CHAPTER 7: FUTURE SCOPE

The Software is currently a prototype level application. The scope and scale of this application can be extended by future developers. The potential areas of extension are listed below.

1. Eradication of the major limitations as listed above.
2. The tool can be made with a GUI rather than a CUI for more ease of use.
3. The tool can be made generic for all type of linux based systems.
4. The tool can be further developed into a generic firewall for a stand alone system which protects the system from all type of intrusions attempts.

CHAPTER 8: CONCLUSION

The project was a great learning experience for both of us and made us learn new concepts of network programming. The project has many potential areas of extensions and with the limitations eradicated the tool can prove really helpful and can be easily used as part of the security mechanism of a system.

APPENDIX

The following selected artifacts form a comprehensive list of the various BSD sockets API structures and functions used in the project.

The sockaddr_in Structure

```
struct sockaddr_in {
sa_family_t sin_family; /* Address Family */
uint16_t sin_port; /* Port number */
struct in_addr sin_addr; /* Internet address */
unsigned char sin_zero[8]; /* Pad bytes */
};
struct in_addr {
uint32_t s_addr; /* Internet address */
};
```

The above structure can be described as follows:

• The sin_family member occupies the same storage area that sa_family does in the generic socket definition. The value of sin_family is initialized to the value of AF_INET.

• The sin_port member defines the TCP/IP port number for the socket address. This value must be in network byte order (this will be elaborated upon later).

• The sin_addr member is defined as the structure in_addr, which holds the IP number in network byte order. If you examine the structure in_addr, you will see that it consists of one 32- bit unsigned integer.

The struct hostent Structure

```
struct hostent {
char *h_name; /* official name of host */
char **h_aliases; /* alias list */
int h_addrtype; /* host address type */
int h_length; /* length of address */
char **h_addr_list; /* list of addresses */
};
/* for backward compatibility */
#define h_addr h_addr_list[0]
```

The hostent h_name Member :

The h_name entry within the hostent structure is the official name of the host that your are looking up. It is also known as the *canonical* name of the host. If you provided an alias, or a hostname without the domain name, then this entry will describe the proper name for what you have queried. This entry is useful for displaying or logging your result to a log file.

The hostent h_aliases Member :

The hostent h_aliases member of the returned structure is an array of alias names for the hostname that you have queried. The end of the list is marked by a NULL pointer.

The recvfrom function

```
z = recvfrom(s, /* Socket */
   dgram, /* Receiving buffer */
   sizeof dgram,/* Max rcv buf size */
   0, /* Flags: no options */
   (struct sockaddr *)&adr, /* Addr */
   &x); /* Addr len, in & out */
```

It is used to receive a datagram packet or to eat a datagram packet.

The getpeername() function

```
int getpeername(int s, struct sockaddr *name, socklen_t *namelen);
```

It is used to retrieve client data from a connected TCP socket.

The gethostbyaddr( ) Function

There are times where you have an Internet address, but you need to report the hostname instead of the IP number. A server might want to log the hostname of the client that has contacted it, instead of the IP number alone. The function synopsis for gethostbyaddr( ) is as follows:

```
#include <sys/socket.h> /* for AF_INET */

struct hostent *gethostbyaddr(
const char *addr, /* Input address */
int len, /* Address length */
int type); /* Address type */
```

The inet_ntoa( ) Function

There are times when a socket address represents the address of a user that has connected to your server, or represents the sender of a UDP packet. The job of converting a network sequenced 32-bit value into dottedquad notation is inconvenient. Hence, the inet_ntoa( ) function has been provided. The synopsis of the function is as follows:

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
char *inet_ntoa(struct in_addr addr);
```

The gethostbyname( ) Function

This function accepts the name of the host that you want to resolve, and it returns a structure identifying it in various ways. The function synopsis is as follows:

```
#include <netdb.h>
extern int h_errno;

struct hostent *gethostbyname(const char *name);
```

The function gethostbyname( ) accepts one input argument that is a C string representing the hostname that you want to resolve into an address. The value returned is a pointer to the hostent structure if the call is successful. If the function fails, then a NULL pointer is returned, and the value of h_errno contains the reason for the failure.

The following selected artifacts form a comprehensive list of the various ncurses functions used in the project which are included in the <ncurses.h> header file.

initscr() Function

The function initscr() initializes the terminal in curses mode. In some implementations it clears the screen and presents a blank screen.

refresh() Function

This function is analogous to normal printf in all respects except that it prints the data in the buffer of a window called stdscr at the current (y,x) co-ordinates.

endwin() Function

This function is used to end the curses mode.

printw() and mvprintw Functions

These two functions work much like printf(). mvprintw() can be used to move the cursor to a position and then print.

attron(ATTRIBUTE) and attroff(ATTRIBUTE) Functions

These two functions switch on and off the attribute passed as an argument respectively.

newwin(height,width,starty,startx) Function

A Window can be created by calling the function newwin( ). It doesn't create any thing on the screen actually. It allocates memory for a structure to manipulate the window and updates the structure with data regarding the window like it's size, starty, startx. The function newwin () returns a pointer to structure WINDOW.

start_color( ) Function

Curses initializes all the colors supported by terminal when start_color () is called.

init_pair()Function

Defines the foreground and background for the pair number given as an argument.

BIBLIOGRAPHY

Books:

Warren W. Gay, Linux Socket Programming by Example, QUE Indiana 2000.

Phil Jones, Knowing Knoppix, Phil Jones California 2005.

Kyle Rankin, Knoppix Hacks, O Reilly .

Manuals:

Pradeep Padla, NCURSES Programming HOWTO

Brian Beej Hall, Beej's Guide to Network Programming

Web References:

http://www.porcupine.org/wietse/

http://en.wikipedia.org/wiki/Knoppix

http://www.knopper.net/

http://www.pjls16812.pwp.blueyonder.co.uk/knowing-knoppix/index.html

www.ecst.csuchico.edu/~beej/