Proceedings of the 4th National Conference; INDIACom-2010
Computing For Nation Development, February 25 – 26, 2010
Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi

# File Transfer Using Secure Sockets in Linux Environment

**Abhipal Singh** abhipal_singh@hotmail.com
**Gurneet Singh Sethi** gurneetsethi@hotmail.com
**Kavleen Kaur Oberoi** kavleenoberoi@gmail.com
**Jasleen Kaur** sippy16@yahoo.com
Guru Tegh Bahadur Institute of Technology, GGSIP University, Delhi

**ABSTRACT**
*File transfer over a network is done in various ways and using different protocols. But a file transferred over a network is not secure. When the network architecture consists of hubs or is wireless then the packets are broadcasted to every computer on that network. In hubs these packets are broadcasted because a hub does not remember what all devices are attached to it. It just takes a packet and sends it on all its ports. Similarly in wireless networks the data packets are broadcasted. In normal circumstances, the computer devices which are supposed to get the file receive it and the other computers reject it. Now, the problem arises when a computer's system is cracked and that computer starts accepting the data it is not supposed to accept. This is the case with packet sniffing. The files sent during a transaction may be captured or "sniffed" by third party tools called "Sniffers". If a sniffer is placed in promiscuous mode, then it can capture all the incoming datagrams and it can reconstruct the original file from those datagrams. To ensure privacy and security we will implement file transfer over Secure Sockets Layer (SSL) so that the file is not interpreted by a sniffer. SSL is a communications protocol layer. It intercepts traffic and provides security between client and the server. Encryption is used to guarantee secure communication in an insecure environment. The encryption/decryption algorithm used for the purpose is Caesar Cipher. First, a file transfer program will be developed using socket programming under Linux environment and then security will be implemented using the above encryption/decryption algorithm.*

**KEYWORDS**
Secured File Transfer, SSL, Cryptography, Caesar Cipher, Socket Programming, Sockets.

## 1.0 INTRODUCTION

This document aims at defining the overall details for 'File Transfer using Secure Sockets Layer'. Its purpose is to transfer a file from one PC to another using encryption. It also shows how unsecured files can be easily sniffed by a simple 'sniffer' application. The intended audiences are all the people who transfer highly confidential data over Internet through TCP/IP layer. This is implemented in Linux environment using C-language. Linux O/S is a preferred choice because it provides the basic framework and header files for implementing socket programming in C. [3][6]

The project is based on a client and server architecture.

## 2.0 THE CLIENT SERVER ARCHITECTURE

In the work, client-server architecture is used to send and receive a file using the sockets made with socket programming. With the code implemented any computer can act as a server or client. Now, the user who wants to send a file is the client and the other computer here acts as a server. The client-server architecture is shown below.

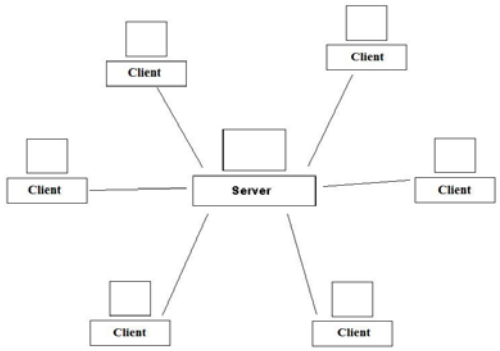



Fig1: client – server architecture

The file which is being transferred is first encrypted and then the file is decrypted by the private key at the server side after the file is received. The figure below shows the secured file transfer [4].

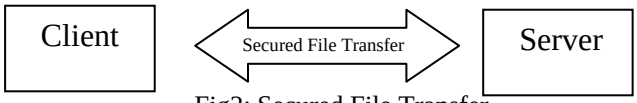



Fig2: Secured File Transfer

## 3.0 SNIFFING OVERVIEW

On a network the data is formatted in packets containing our information and communication overhead information.
The computer contains one or several network interface cards. Each of those cards has a unique MAC address and IP address. When two people establish a connection, each packet contains the address of the two network cards, and those addresses are used to route the packet to the correct PC. But a PC connected to a network sees all the data packets travelling on the wire. In the usual case, only those packets with the NIC address of one of our cards are fetched and sent to the software.

Now the case not only here makes the sniffer able to see all the packets, but also to jump on the traffic without participating

directly to any conversation. This is what sniffing is all about [3].

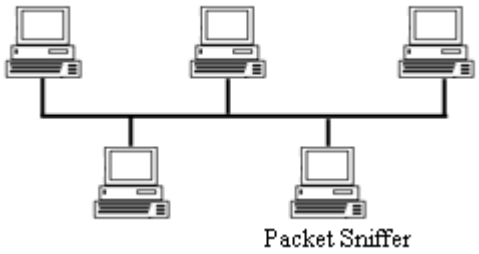



Fig3: Packet Sniffer on network

**4.0 SOCKET PROGRAMMING – SOCKETS**

In this work we have first created a program to send and receive a file through sockets in the linux environment. A socket is a point for communication which is identified by the machine's IP address and the protocol port number. It is an Application Programming Interface to access transport protocols of an operating system.

Processes on different/same computers communicate with each other through sockets. For example, sockets can be used for Unix inter process communication. Unlike FIFOs, sockets are bidirectional. This forms the basis of much of Internet networking. [6]

**Advantages of sockets in C/C++ over Java:**

- Sockets in C/C++ are much faster than any other language such as java. Since C/C++ has much lower abstraction level therefore it provides much more functionality.
- The functionality is more in case of C/C++ because the things which are done automatically in languages like java have to be done manually with C/C++. This gives the programmer a better insight of the code and its functionality.
- For example: Functions such as connect(), listen() and bind() are not explicitly used in java.

**Other Advantages of Sockets**

- Computers on different platforms can also communicate through sockets.
- Same procedure and complexity is followed whether the communication is done on the same computer or on far away places.
- Sockets use file descriptors, so many standard Unix file handling calls can be used. Like: read( ), write( ), close( ) etc.

A call to function socket() with appropriate arguments is made to create a Unix socket. A Unix socket can be of two types as follows:

1) Unix internal (local) sockets

Similar to FIFOs - can only be used between processes running on the same machine.

2) Unix network (internet) sockets

Communication between any two processes on any machines that are networked is done.

Sockets are so called because one socket is "plugged into" another socket for communication over a network [6] [7].

**5.0 TYPES OF SOCKETS**

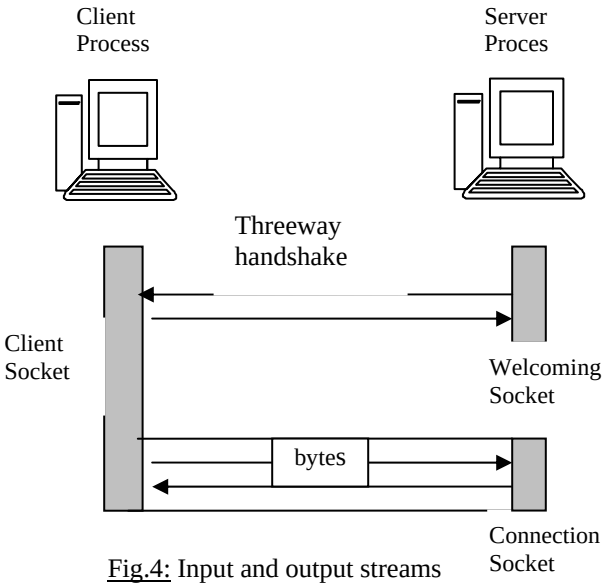



Fig.4: Input and output streams

a) Stream Sockets

- These sockets use TCP (Transmission Control Protocol), which is a reliable, stream oriented protocol where streams are used for input and output data.
- The client creates a TCP socket by specifying the server's IP address and port number of the server process.
- When client creates a TCP socket it establishes a passive connection with the server TCP.
- Server TCP creates a new TCP connection for each client that requests a communication so that it can handle the communication with multiple clients at the same time.
- Once a client server control connection is created, data can be transferred as long as that control connection is active.

b) Datagram sockets

- These sockets use UDP (Unix Datagram Protocol), which is unreliable and message oriented where datagrams are used for input and output data.
- No handshaking is done with UDP sockets since the client explicitly attaches IP address and port of destination (i.e. server).
- Transmitted data may be received out of order or lost completely.

- This type of packet can be sent at any time to the destination.
- Since they are connectionless, therefore they are not globally unique.
- These are generally smaller than TCP sockets.

c) Raw IP sockets

- These types of sockets are used to create raw IP Packets bypassing the transport layer.
- The type SOCK_RAW is used when socket() is called.
- The packet is directly passed to the application that needs it without making it to go through the whole encapsulation/ decapsulation process.
- The headers are made by the process retrieving the data from the sockets rather than the complex TCP/IP mechanism.
- These are not used in java [6] [7].

## 6.0 TCP/IP CLIENT SERVER MODEL FOR SOCKET PROGRAMMING

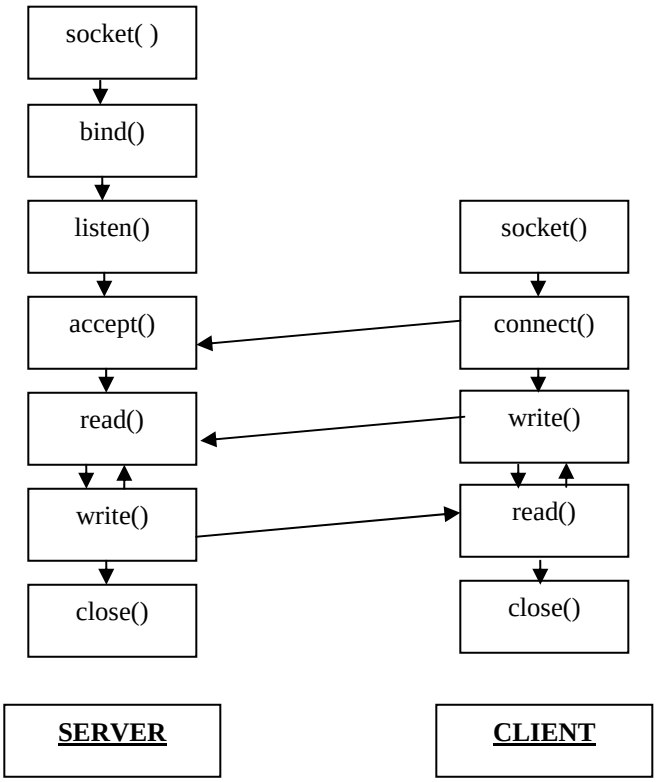



Fig5: Client-Server model for socket programming

The software which is developed to implement the file transfer, first needs the server to open a socket and listen for connection from clients. After receiving an active connection from a client who wants to send a file to the server, it accepts the file and then closes the socket used in the session [7].

## 7.0 CAESAR CIPHER

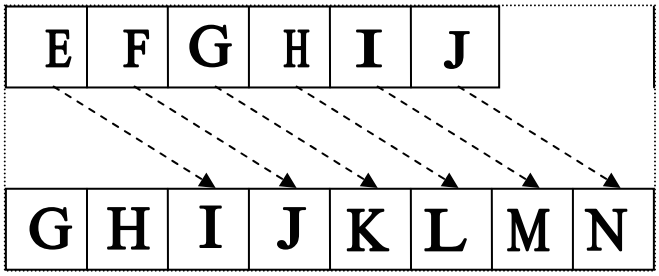


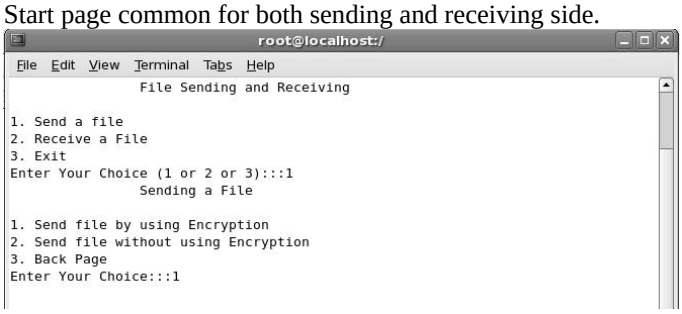

Fig6: Caesar Cipher

In cryptography, a Caesar cipher, also known as the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet [1] [5].

## 8.0 SCREEN SHOTS

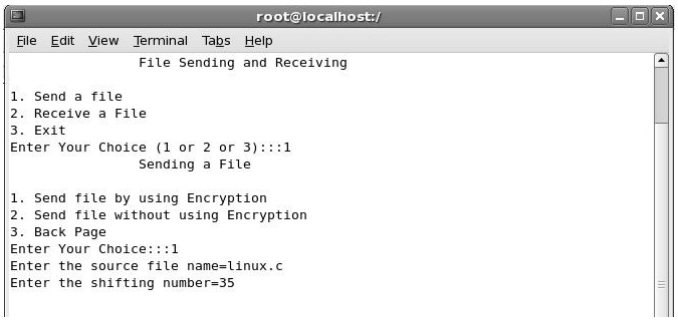
**At Client Side for Sending the File**


Start page common for both sending and receiving side.

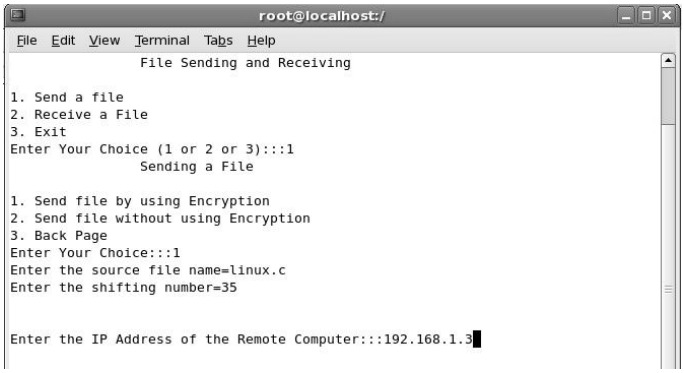



Input your choice for sending file with or without encryption.


Enter the name of the file to send

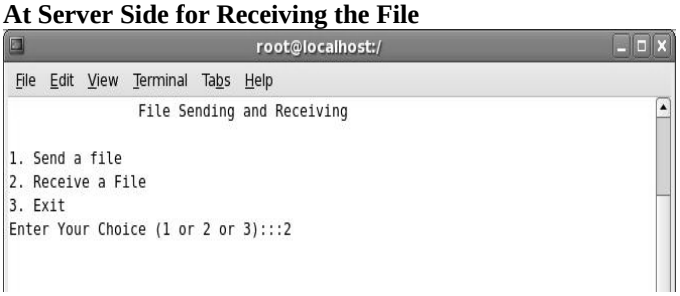



Enter the shifting number for the caesar cipher to encrypt data.

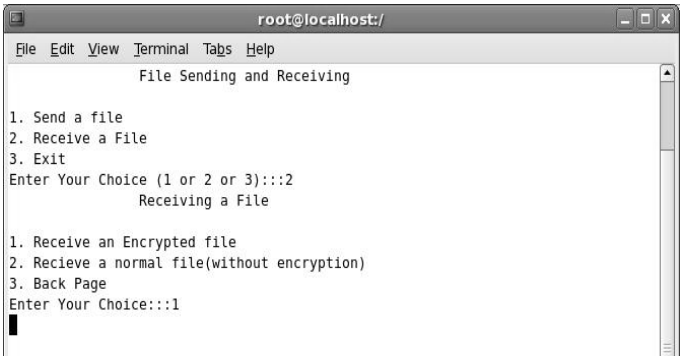



Enter the IP address of the server machine where the file is to be received.

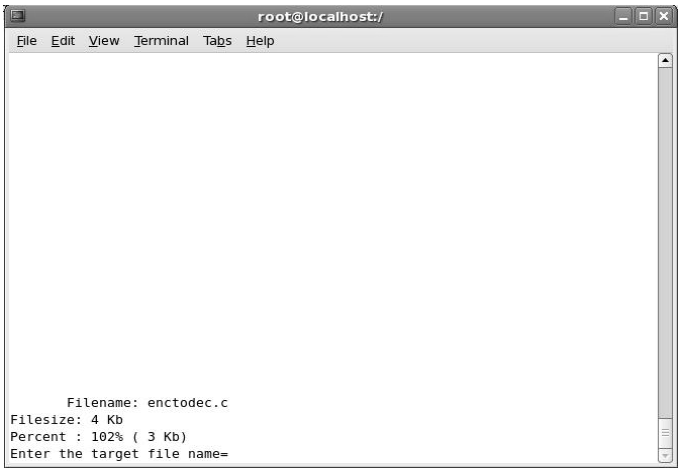
**At Server Side for Receiving the File**




Start page common for both sending and receiving side.




Input your choice to receive an encrypted or normal file




Enter the name by which you want to save the received file


Enter the shifting number or the private key to decrypt the file

## 9.0 CONCLUSION

Data transfer over a network with hubs or a wireless network is broadcasted and therefore the data is reached to all devices connected on that network. The computer device to which this data is intended accepts the data and other devices rejects the data, but the devices on which special packet sniffers are employed they also start accepting this data and then the transfer becomes unsecure.

To make this transaction secured we encrypted the file and then send it through the socket program code. In this case even if a packet sniffer tries to accept this transaction, it will only get "garbage" content and not the exact transaction contents.

## 10.0 FUTURE SCOPE

The file transfer in a network can be made even more secured if we implement the Public Key Infrastructure (PKI). The PKI is an architecture introduced to increase the level of security for the data exchanged over a network.

*Continued on Page No. 144*

***Continued from Page No. 140***

It uses a mathematical technique called public key cryptography which uses a pair of related cryptographic keys in order to verify the identity of the sender (through signing) and/or to ensure privacy (through encryption of data).
In this way when we have a primary and public key we can further enhance privacy [2].

**11.0 REFERENCES**

[1] http://en.wikipedia.org/wiki/Caesar_cipher
[2] Mohsen Toorani, and Ali Asghar Beheshti Shirazi, "LPKI - A Lightweight Public Key Infrastructure for the Mobile Environments", Proceedings of the 11th IEEE International Conference on Communication Systems (IEEE ICCS'08), Guangzhou, China, Nov. 2008.
[3] Felix John COLIBRI, “TCP IP Sniffer” http://www.felix-colibri.com/papers/colibri_utilities/tcp_ip_sniffer/tcp_ip_sniffer.html'
[4] Marin, G.A., “Network Security Basics”, Security and Privacy, IEEE, Volume -3 Nov-Dec 2005
[5] William Stallings, “Cryptography and Network Security, Third Edition”.
[6] Gary R. Wright, W. Richard Stevens, “TCP/IP Illustrated, Volume 2: The Implementation”.
[7] IBM -iSeries Information Center, Version 5 Release 3, http://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp?topic=/rzab6/rzab6uafunix.htm