International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org
Volume 4, Issue 3, March 2015 **ISSN 2319 - 4847**

SECURE SHELL- ITS SIGNIFICANCE IN NETWORKING (SSH)

ANOOSHA GARIMELLA , D.RAKESH KUMAR

[1.] B. TECH, COMPUTER SCIENCE AND ENGINEERING Student, 3rd year-2nd Semester
GITAM UNIVERSITY Visakhapatnam, Andhra Pradesh India

[2.]Assistant Professor Computer Science and Engineering
GITAM UNIVERSITY Visakhapatnam, Andhra Pradesh India

ABSTRACT

This paper is focused on the evolution of SSH, the need for SSH, working of SSH, its major components and features of SSH. As the number of users over the Internet is increasing, there is a greater threat of your data being vulnerable. Secure Shell (SSH) Protocol provides a secure method for remote login and other secure network services over an insecure network. The SSH protocol has been designed to support many features along with proper security. This architecture with the help of its inbuilt layers which are independent of each other provides user authentication, integrity, and confidentiality, connection-oriented end to end delivery, multiplexes encrypted tunnel into several logical channels, provides datagram delivery across multiple networks and may optionally provide compression. Here, we have also described in detail what every layer of the architecture does along with the connection establishment. Some of the threats which Ssh can encounter, applications, advantages and disadvantages have also been mentioned in this document.

Keywords: SSH, Cryptography, Port Forwarding, Secure SSH Tunnel, Key Exchange, IP spoofing, Connection-Hijacking.

1. INTRODUCTION

SSH Secure Shell was first created in 1995 by Tatu Ylonen with the release of version 1.0 of SSH Secure Shell and the Internet Draft "The SSH Secure Shell Remote Login Protocol". SSH functions as a type of tunnel for encoding login procedures. The encryption used by SSH is intended to provide confidentiality and integrity of data over an unsecured network, such as the Internet There are several ways to use SSH. One way is to use the automatically generated public-private key pairs to simply encrypt a network connection, and then use password authentication to log on. Another way is to use a manually generated public-private key pair to perform the authentication, allowing users to log in without having to specify a password. Here, anyone can produce a matching pair of different keys; the keys can be either public or private. The public key is placed on all computers that must allow access to the owner of the matching private key while the owner keeps the private key secret. The actual authentication is based on the private key; the key itself is never transferred through the network during authentication. SSH only verifies whether the same person having the public key is also owning the private key. It is essential to verify all unknown public keys and associate the public keys with identities of the private key before accepting them as valid. Public key without validation will authorize an unauthorized attacker as a valid user.

2. OVERVIEW OF SSH

SSH Secure Shell is a UNIX based command interface and a cryptographic network protocol used to protect data in transmission between devices providing strong authentication and establishes a secure channel over an insecure network in a client-server architecture, connecting an SSH client application with an SSH server. Secure Shell is a program to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another. It is comprised of a suite of three utilities, Slogin, Ssh and SCP. Other basic utilities include Ssh-add, ssh-agent, ssh-keysign, ssh-keyscan, ssh-keygen, and sftp-server. It is a replacement for earlier versions of UNIX utilities rlogin, rsh, rcp, and rdist. The use of these SSH Secure Shell utilities at both ends of the connection are authenticated using a digital certificate. Unlike utilities such as Telnet, the passwords are encrypted. This encryption makes it difficult for someone to breach the confidentiality of your data or compromise passwords. It can be implemented over most operating systems (Win, MAC, and Unix/Linux). SSH Secure Shell uses RSA public key cryptography for connection and authentication. These encryption algorithms include DES, 3DES, Blowfish and IDEA to authenticate both the ends of the connection, encrypt all transmitted data, confidentiality, data compression, protect data integrity, multiplexing with ssh-2 and validate values returned by services such as DNS or network protocols

(TCP). It is commonly used for controlling web, application servers and network appliances remotely. It is not a shell or a command interpreter. It is only a channel to run a shell over a remote computer. SSH is also not a complete security solution. It won't protect computers from active break-in attempts or denial-of-service attacks, and it won't eliminate other hazards such as viruses, Trojan horses, and coffee spills. It does, however, provide robust and user-friendly encryption and authentication.

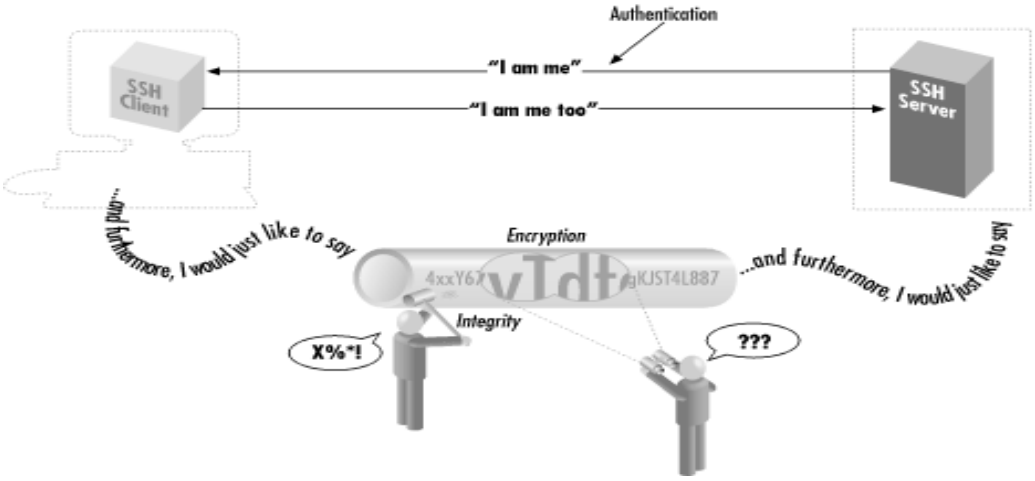

Figure 1: illustrates the authentication, encryption and integrity using SSH

3. WORKING OF SSH

Ssh works by the exchange and verification of information, using public and private keys, to identify hosts and users. It provides encryption of subsequent communication, also by the use of public/private key cryptography. Client refers to a workstation or PC that you are already logged in to, e.g., your own personal workstation or a group workstation that provides XDM session management for several X terminals. The term server means a secondary remote workstation that you wish to log in to do some work for example, a login session server. The client is where you type "rlogin server" or "rcp file server: newfile" and the server is where you get a new login session and shell prompt or are copying files. A user can generate an identity in the client system by running the ssh-keygen program. This program creates a subdirectory $HOME/.ssh and inserts in it two files named identity and identity.pub which contain your private and public keys for your account on the client system. The latter file can then be appended to a file $HOME/.ssh/authorized_keys that should reside on all servers where you will make Ssh connections. As a system administrator, you generate a public and private key pair for the system itself. By use of this information contained within the system itself, the possibility of someone spoofing the system's identity by faking IP addresses or mugging up DNS records that associate IP addresses and domain names is removed.

4. NEED FOR SSH

With the evolution of the internet, the number of threats is also increasing where some of the threats are network monitoring, connection hijacking, routing spoofing, DNS spoofing, denial of service attacks, etc. File transfers, remote logins, and remote command executions became possible with the help of existing protocols like ftp, telnet and tcp but the problem existed with these protocols is they lacked security(r-commands) and hence it is possible for an intruder to intercept and read the data. Telnet especially was very risky as the username and password were interpreted as plain text over the network. A protocol was needed to fix these issues and then Ssh had come into picture.

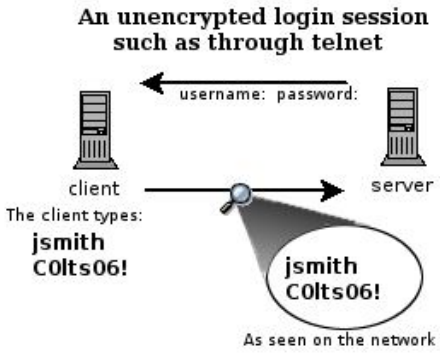

Figure 2: illustrates how username and passwords are seen over a network when telnet is used.

The diagram on the top shows how a telnet session can be viewed by anyone on the network by using a sniffing program like Ethereal or tcpdump. Anyone on the network can steal your passwords and other information. The first diagram shows user Jsmith logging in to a remote server through a telnet connection. He types his username Jsmith and password C0lts06! which are visible to anyone who is using the same networks that he is using.

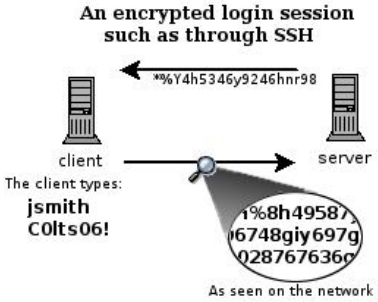

Figure 3: illustrates how data is encrypted over a network using SSH

The second diagram shows how the data in an encrypted connection like SSH is encrypted on the network and so cannot be read by anyone who doesn't have the session keys .The server still can read the information, but only after negotiating the encrypted session with the client. As in the above case password of Jsmith is not being shown as clear text on the network.

5. MAJOR SSH COMPONENTS/ ARCHITECTURE

SSH is organized as three protocols that typically run on top of TCP

- Transport Layer Protocol: Provides server authentication, data confidentiality, and data integrity with forward secrecy. The transport layer may optionally provide data compression.
- User Authentication Protocol: Authenticates the user to the server
- Connection Protocol: Multiplexes multiple logical communications channels over a single underlying SSH connection.

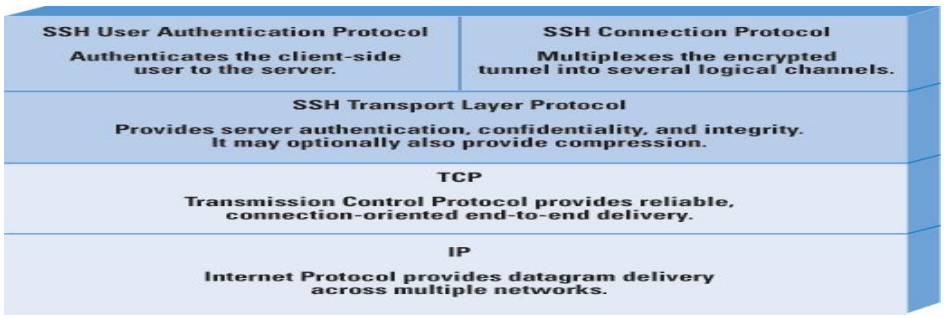

Fig 4: Architecture of SSH

5.1 Transport Layer Protocol

Server authentication occurs at the transport layer, based on the server possessing a public-private key pair. A server may have multiple host keys using multiple different asymmetric encryption algorithms. Multiple hosts may share the same host key. In any case, the server host key is used during key exchange to authenticate the identity of the host. For this authentication to be possible, the client must have presumptive knowledge of the server public host key. As per RFC 4251 two alternative trust models can be used:

a) The client has a local database that associates each host name with the corresponding public host key. This method requires no centrally administered infrastructure and no third-party coordination. The disadvantage is that the database of name-to-key associations may become burdensome to maintain.

b) The host name-to-key association is certified by a trusted Certification Authority (CA). The client knows only the CA root key and can verify the validity of all host keys certified by accepted CAs. This alternative eases the maintenance problem, because ideally only a single CA key needs to be securely stored on the client. On the other hand, each host key must be appropriately certified by a central authority before authorization is possible.

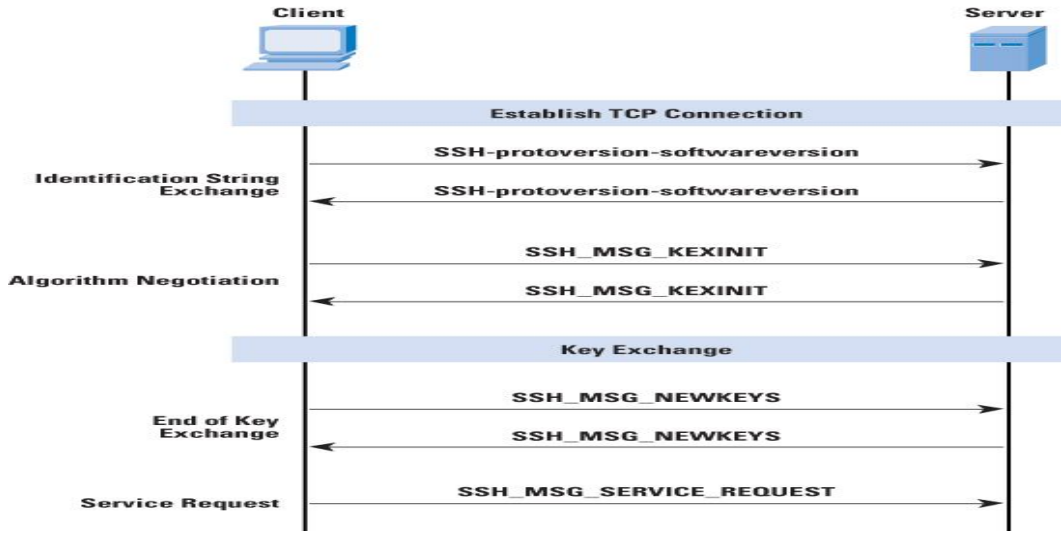

Figure 5: illustrates the sequence of events in the SSH Transport Layer Protocol.

First, the client establishes a TCP connection to the server with the TCP protocol and is not part of the Transport Layer Protocol. When the connection is established, the client and server exchange data, referred to as packets, in the data field of a TCP segment. Each packet is in the following format.

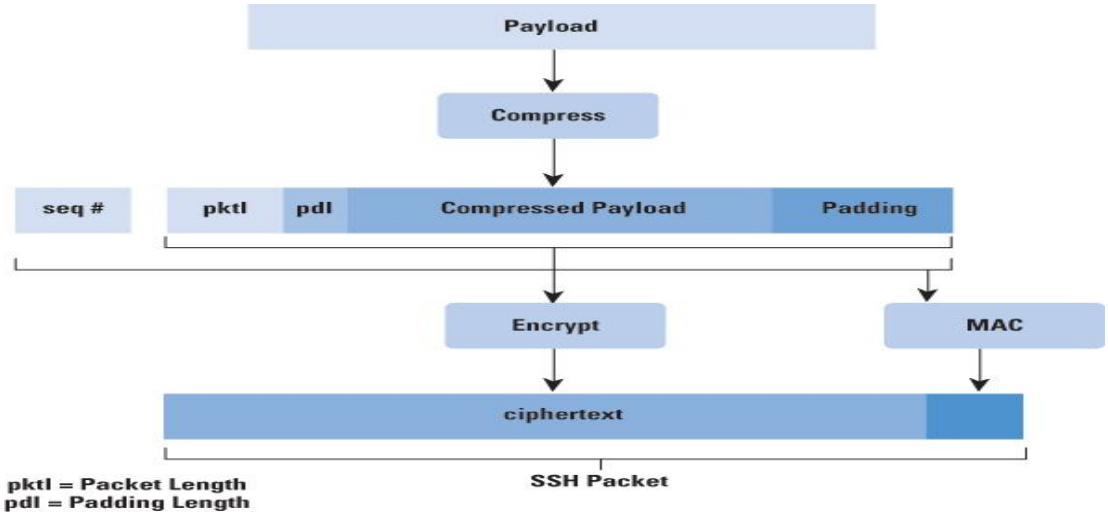

Figure 6: shows the packet format that is followed during transmission

Packet length: Packet length is the length of the packet in bytes, not including the packet length and Message Authentication Code (MAC) fields. Padding length: Padding length is the length of the random padding field. Payload: Payload constitutes the useful contents of the packet. Prior to algorithm negotiation, this field is uncompressed. If compression is negotiated, then in subsequent packets this field is compressed. Random padding: After an encryption algorithm is negotiated, this field is added. It contains random bytes of padding so that that total length of the packet (excluding the MAC field) is a multiple of the cipher block size, or 8 bytes for a stream cipher. Message Authentication Code (MAC): If message authentication has been negotiated, this field contains the MAC value. The MAC value is computed over the entire packet plus a sequence number, excluding the MAC field. The sequence number is an implicit 32-bit packet sequence that is initialized to zero for the first packet and incremented for every packet. The sequence number is not included in the packet sent over the TCP connection.

5.2 User Authentication Protocol

The User Authentication Protocol provides the means by which the client is authenticated to the server. Three types of messages are always used in the User Authentication Protocol. Authentication requests from the client have the format:

Byte SSH_MSG_USERAUTH_REQUEST (50)
String Username
String Service name
String Method name
… Method specific fields

Where username is the authorization identity the client is claiming, service name is the facility to which the client is requesting access and method name is the authentication method being used in this request. The first byte has decimal value 50 which is interpreted as SSH_MSG_ USERAUTH_REQUEST. The server either accepts or rejects the request but requires one or more authentication methods.

Byte SSH_MSG_USERAUTH_FAILURE (52)
Name-list Authentications that can continue
Boolean Partial Success

Where name list is a list of methods. If the server accepts authentication, it sends a single byte message SSH_MSG_USERAUTH_SUCCESS (52).

The exchange of message involves the following steps:

The client sends a message SSH_MESSAGE_USERAUTH_FAILURE. The server checks if the username is valid, if the username is valid, it proceeds to the third step or else the server returns SSH_MSG_USERAUTH_FAILURE with value false. The server sends a list of methods for the client to be authenticated. The client selects one of the acceptable method names and sends a message SSH_MSG_USERAUTH_REQUEST with the required method parameters. When all required authentication succeeds, the server sends a SSH_MSG_USERAUTH_SUCCESS message and the authentication protocol is over.

The server may require one or more of the following authentication methods:

Public key: The details of this method depend on the public-key algorithm chosen. In essence, the client sends a message to the server that contains the client's public key, with the message signed by the client's private key. When the server receives this message, it checks to see whether the supplied key is acceptable for authentication and, if so, it checks to see whether the signature is correct.

Password: The client sends a message containing a plaintext password, which is protected by encryption by the transport Layer Protocol.

Host based: Authentication is performed on the client's host rather than the client itself. Thus, a host that supports multiple clients would provide authentication for all its clients. This method works by having the client send a signature created with the private key of the client host. Thus, rather than directly verifying the user's identity, the SSH server verifies the identity of the client host—and then believes the host when it says the user has already authenticated on the client side.

5.3 Connection Protocol

The SSH Connection Protocol runs on top of the SSH Transport Layer Protocol and assumes that a secure authentication connection is in use. The secure authentication connection, referred to as a tunnel, is used by the Connection Protocol to multiplex a number of logical channels. RFC 4254, "The Secure Shell (SSH) Connection Protocol," states that the Connection Protocol runs on top of the Transport Layer Protocol and the User Authentication Protocol. RFC 4251, "SSH Protocol Architecture," states that the Connection Protocol runs over the User Authentication Protocol. In fact, the Connection Protocol runs over the Transport Layer Protocol, but assumes that the User Authentication Protocol has been previously invoked.

All types of communication using SSH, such as a terminal session, are supported using separate channels. Either side may open a channel. For each channel, each side associates a unique channel number, which need not be the same on both ends. Channels are flow-controlled using a window mechanism. No data may be sent to a channel until a message is received to indicate that window space is available. The life of a channel progresses through three stages: opening a channel, data transfer, and closing a channel.

When either side wishes to open a new channel, it allocates a local number for the channel and then sends a message of the form:

Byte SSH_MESSAGE_CHANNEL_OPEN
String Channel Type
Unit 32 Sender Channel
Unit 32 Initial Window Size
Unit 32 Maximum Packet Size

Where 32 is 32-bit unsigned integer. The channel type identifies the application for this channel. The sender channel is the local channel number. The initial window size specifies how many bytes of channel data can be sent to the sender of this message without adjusting the window. The maximum packet size specifies the maximum size of an individual data packet that can be sent to the sender.

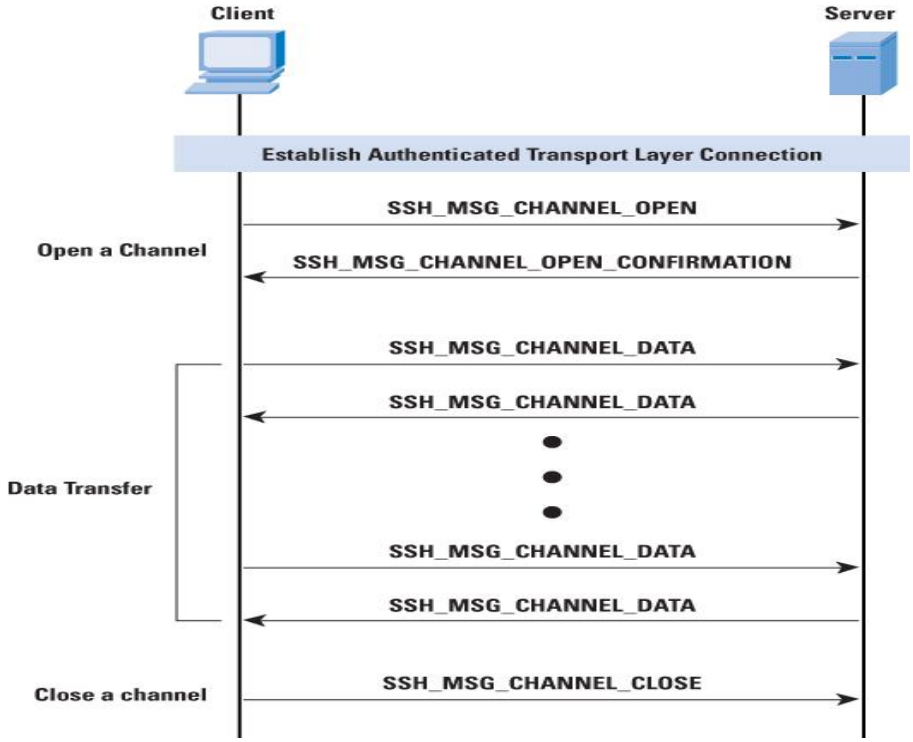

Figure 7: illustrates the data transfer through a channel

Four channel types are recognized in the SSH Connection Protocol specification:

Session: Session refers to the remote execution of a program. The program may be a shell, an application such as file transfer or e-mail, a system command, or some built-in subsystem. When a session channel is opened, subsequent requests are used to start the remote program.

X11: This channel type refers to the X Window System, a computer software system and network protocol that provides a GUI for networked computers. X allows applications to run on a network server but be displayed on a desktop machine.

Forwarded-tcpip: This channel type is remote port forwarding.

Direct-tcpip: This channel type is local port forwarding.

6. SSH FEATURES

6.1 Secure Remote Logins

Suppose you have an account on another computer over the Internet. Now, you will connect from a home PC to an ISP and use a telnet program to log in to your accounts on other computers. Your entire log in process is visible over the network. Ssh overcomes such insecure problems. The client authenticates you to the remote computers SSH server using an encrypted connection, meaning that your username and password are encrypted before they leave the local machine. The SSH server then logs you in, and your entire login session is encrypted as it travels between client and server.

6.2 Secure File Transfer

Suppose you have two accounts on two different computers over the Internet. You want to transfer a file from one computer to the other. The file that is to be transferred has confidential information that should not be accessed by a third party. Under such circumstances, you can use the following command to transfer the file securely between machines with a single secure copy command which encrypts the file as it leaves and decrypts at the destination.

$ SCP my_file secure@xyz.com;

6.3 Keys and Agents

Suppose you have accounts on many computers on a network. For security reasons, you prefer different passwords on all accounts; but remembering so many passwords is difficult. More the number of times you try higher are the chances of making mistakes. It's also a security problem in itself. Thus, Ssh provides a mechanism to overcome this. You can identify all your accounts at a single go without typing your passwords continually. Rather than passwords keys are used here. Keys are encrypted and it may be used only after a secret phrase is used to decrypt it. The working is as follows:

a) Place all special files called public key files into your remote computer accounts. These enable you're SSH clients to access your remote accounts.

b) On your local machine, invoke the ssh-agent program, which runs in the background.

c) Choose the key (or keys) you will need during your login session.

d) Load the keys into the agent with the Ssh-add program. This requires knowledge of each key's secret passphrase.

6.4 Access Control

Suppose you want another person to access your account but only for certain purposes. For example, while you're out of town you'd like your friend to read your email but not to do anything else in your account. With SSH, you can give your friend access to your account without revealing or changing your password, and with only the ability to run the email program not anything else.

6.5 Port Forwarding

One of the most useful features of SSH is port forwarding. Port forwarding provides the ability to convert any insecure TCP connection into a secure SSH connection. It is also referred to as SSH tunneling. We need to know what a port is in this context. A port is an identifier of a user of TCP. So, any application that runs on top of TCP has a port number. Incoming TCP traffic is delivered to the appropriate application on the basis of the port number. An application may employ multiple port numbers. For example, for the Simple Mail Transfer Protocol (SMTP), the server side generally listens on port 25, so that an incoming SMTP request uses TCP and addresses the data to destination port 25. TCP recognizes that this address is the SMTP server address and routes the data to the SMTP server application.

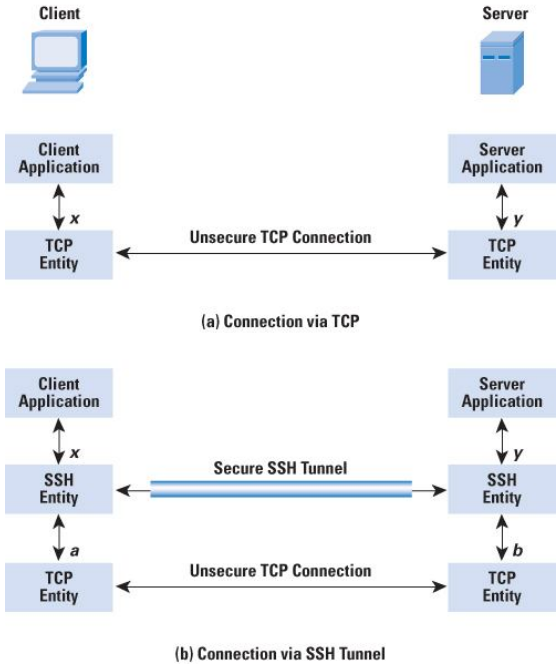

Figure 8 illustrates port forwarding

We have a client application that is identified by port number x and a server application identified by port number y. At some point, the client application invokes the local TCP entity and requests a connection to the remote server on port y. The local TCP entity negotiates a TCP connection with the remote TCP entity, such that the connection links local port x to remote port y. To secure this connection, SSH is configured so that the SSH Transport Layer Protocol establishes a TCP connection between the SSH client and server entities with TCP port numbers 'a' and 'b' respectively. A secure SSH tunnel is established over this TCP connection. Traffic from the client at port x is redirected to the local SSH entity and travels through the tunnel where the remote SSH entity delivers the data to the server application on port y. Traffic in the other direction is similarly redirected.

6.6 Secure remote command execution

A System administrator needs to run the same command on many computers. You'd like to view the active processes for each user on four different computers group1, group2, group3 and group4 on a local area network using the UNIX command /usr/ucb/w.

```
#!/bin/sh
For machine in group2
do
rsh $machine/usr/ucb/w
done
```

The results of usr/ucb/w are displayed as plain text across the network. If the information is sensitive then this is unacceptable. Instead we can use Ssh in place of rsh then the commands and its results are encrypted as they travel across the network, and strong authentication techniques may be used when connecting to the remote machines. (Use the command Ssh in place of rsh).

7. THREATS SSH CAN ENCOUNTER

7.1 Eavesdropping

An eavesdropper is an attacker who reads network traffic without affecting it in any way. SSH's encryption prevents eavesdropping. The contents of an SSH session, even if intercepted, can't be decrypted by an attacker.

7.2 Name Service and IP Spoofing

If an attacker diverts your naming service, network-related programs may be coerced to connect to the wrong machine. Similarly, an attacker can impersonate a host by stealing use of its IP addresses. Your client program can connect to a false server that steals your password when you supply it. SSH guards against this attack by cryptographically verifying the server host identity. When setting up a session, the SSH client validates the server's host key against a local list associating server names and addresses with their keys. If the supplied host key doesn't match the one on the list, SSH complains. The SSH-2 protocol allows for including PKI certificates along with keys.

7.3 Connection Hijacking

An attacker can hijack a TCP connection. The attacker can simply wait until you've logged in, then steal your connection and insert his own commands into your session. SSH can't prevent hijacking, since this is a weakness in TCP, which operates below SSH. However, SSH renders it ineffective. SSH's integrity checking detects if a session is modified in transit, and shuts down the connection immediately without using any of the corrupted data.

7.4 Men-in-the-Middle Attacks

Imagine that you try to connect to an SSH server, but an attacker intercepts your connection. He behaves just like an SSH server, though, so you don't notice, and she ends up sharing a session key with you. Simultaneously, he also initiates her own connection to your intended server, obtaining a separate session key with the server. He can now log in as you because you used password authentication has handed him your password. You and the server both think you have a connection to each other, when in fact you both have connections to a third person instead. All the data can now be passed data back and forth between you and the server. The data can be read as well as be modified.

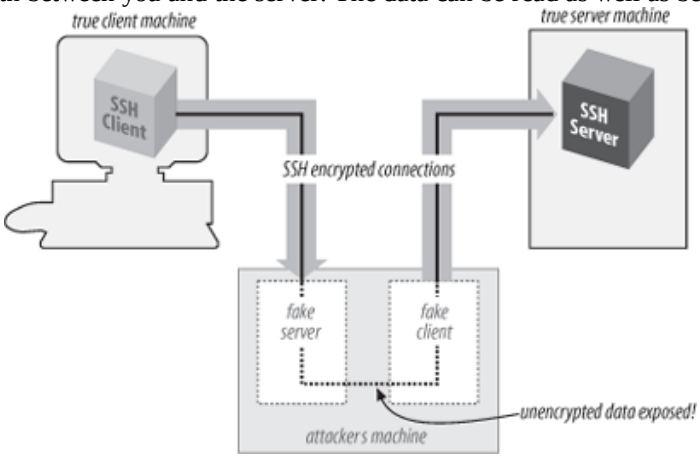

Figure9 Man-in-the-middle attack

SSH prevents such attacks in two ways:

1) The first is server host authentication. Unless the attacker has the server's private key he cannot effect in any way. For this protection to work, it is crucial that the client checks the server-supplied public host key against its known hosts list. If you connect for the first time to a new server and let Ssh accept the host key, you are actually open to a man-in-the-middle attack.
2) The second protection SSH affords is via certain user authentication methods. The password method is obviously vulnerable, but public key and host based authentication resist man in the middle attacks. He can't discover the session key simply by observing the key exchange; he must perform an active attack in which he carries out separate exchanges with each side, obtaining separate keys of his own with the client and server. When a client provides a digital signature for either public-key or host based authentication, it includes the session identifier in the data signed. Thus, he cannot authenticate the client-server connection. If you don't verify the server name/key authentication, he can still perform the man-in-the-middle attack, even though he can't log in as you on the server side.

8. APPLICATIONS

- Back-ups can be run over Ssh
- Ssh can be used to communicate across a firewall
- Secure Shell can be used along with TCP wrappers for added security
- Ssh can be used with rsync or rdist.

9. ADVANTAGES

- **User-Authentication:**

Authentication is the means by which a system verifies that access is only given to intended users and denied to anyone else. Most Secure Shell implementations include password and public key authentication methods. The Secure Shell protocol's flexibility allows new authentication methods to be incorporated into the system as they become available.

- **Password Authentication:**

Username and Password are a common way of authenticating one user. Protocols like telnet display the username and password clearly as ASCII text making them less secure. However, Secure Shells makes sure that all the usernames and passwords are encrypted before they pass onto the network. These user name and password authentication mechanisms prevent eavesdropping by attackers who would otherwise trap sensitive data.

- **Prevents IP source routing**

While source routing is normally used for good purposes such as altering the path of a network signal if it originally fails, it can also be used by malicious users on to make a machine think it is talking to a different one. Using secure shell the malicious use of IP source routing can be avoided.

- **Keeps us safe from DNS Spoofing**

This is a type of hacking attack where data is inserted into a Domain Name System name server's cache database. This causes the name server to return an incorrect IP address so it can divert traffic to another computer. This is often the attacker's computer and from there they can obtain sensitive information. As Secure Shell uses the techniques of encryption and key exchange, the chances of diversion of an IP Address are less.

- **Data manipulation at routers along the network cannot be done**

Attacker obtains or changes data at intermediaries along the network route. This is often performed at routers where data enters a sort of gateway or checkpoint on the way to its destination. As the data is encrypted, it only appears in the form of some random characters, so tracking of data by intruders would be difficult.

- **IP address spoofing**

IP spoofing is the when a malicious user creates packets with a forged source IP address. This way it keeps the source computer's identity and location a secret and appears to be another computer that the receiver trusts. Though the packets are forged, the private key prevents the data packet from being sent to a wrong destination.

- **Easy to manage a dedicated server remotely.**

With SSH you can manage your dedicated server remotely, monitor logs, install applications, start and stop services, and even manipulate databases. It recognizes normal UNIX commands, and you can use it to login as root for full system administration as SSH is the most secure and reliable way to manage your server.

10. DISADVANTAGES

- Port ranges & dynamic ports can't be forwarded.
- SSH server daemon problem.
  We cannot restrict what ports may or may not be forwarded, per user. When a user is authenticated by password, the client's RSA identity is not verified against Ssh known hosts.
- Performance is not too high on old machines.
- The standard SSH1 distribution's defaults include a clear text option and patented algorithms such as IDEA.
- Licensing of the original source has become very restrictive
- Port forwarding can also introduce security problems, is not used correctly. The SSH server doesn't allow detailed configuration of what forwarding is allowed from which client to which server.

11. CONCLUSION

By the end of this paper, it is concluded that using Secure Shell is one of the solutions for the prevailing security concerns over the network. Secondly, not only secure file transfers, it also helps in remote login, port forwarding and other access control mechanisms. It not only removes the risk of security but also gives us a scope for developing larger networks with advanced features. Though Ssh has its disadvantages and threats like men in the middle attacks, performance, port problems and other restrictions, further research is being aimed to improve these and add new features. The number of growing concerns for sensitive data over the networks is leading to further expansion of network security.

REFERENCES

[1] Ylonen, T. and Lonvick, C., The Secure Shell (SSH) Protocol Architecture, RFC 4251, January 2006.
[2] Daniel J. Barrett, Richard E. Silverman (2001), SSH, the Secure Shell: The Definitive Guide, O'Reilly Media.
[3] William Stallings, Network Security Essentials, fourth edition, Prentice Hall, 432 pp.
[4] Michael W Lucas SSH Mastery, openSsh, putty, tunnels and keys, Tilted Windmill Press.

[5] Omar Santos, End-to-End network Security, CISCO PRESS, 2008.
[6] John. E. Canavan, Fundamentals of network Security, 1999.
[7] William Stallings, Network Security Essentials, fourth edition, Prentice Hall, 432 pp. (Figures 4,5,6,7,8)

AUTHOR

I'm a student pursuing B.Tech in Computer Science and Engineering from GITAM UNIVERSITY, Visakhapatnam. I have a strong academic background with aggregates above 90% throughout. Research is my key interest and Network Security is the area I have chosen for research. The above paper is a part of my research on Secure Shell under the guidance of my professor Sri. D. Rakesh Kumar working as an assistant professor in GITAM UNIVERSITY, Visakhapatnam.