

The Evolution of SSH

¹ Amit Gupta, ² Ajay Goel, ³ Surbhi Gupta

¹ Deptt. Of Electronics & Communication, Swami Devi Dyal Institute of Engineering & Technology, Barwala

² Deptt. Of Computer Sc. & Engg., Kurukshetra Institute of technology & management. Kurukshetra – 136119

³ Deptt. Of Electrical Engg., RIMT, Mandigobindgarh

Email: ¹goel_ajay1@yahoo.com, ²amitgupta45@hotmail.com

Abstract- The menace to an enterprise systems and data continue to rise at a frightening rate. It's still relatively easy for hackers to break into corporate or government networks and cause irrevocable damage. The risks aren't limited to data exposure, but also include threats to data integrity, access, and authentication. To keep data secure is a small, robust, easy-to-configure software solution called Secure Shell, or SSH. SSH is typically a set of tools based on the SSH protocol. The outlook is that a large organization with a diverse network environment, consisting of a large number of servers that run with different operating systems. The Secure Shell (SSH) is a more secure technology in the mixed environments for the network security. This paper addresses the need of that domain of users, especially, enterprisers and home users who need to safe and sound data instead of FTP, Telnet and other system access methods. The SSH protocol Architecture, features, versions including the latest SSH Tectia are discussed in this paper. This Paper describes common vulnerabilities and known exploits of plaintext file transfer and the terminal connections. We describe the major threats and attacks that can compromises the network security and how to eradicate them. We perform an analysis which shows that how and why SSH technology succeeded in network security.

Index Terms: FTP, Telnet, Open SSH, SSH1, SSH2, SSH Tectia

I. INTRODUCTION

Sometimes, in certain cases, the data is transmitted across the networks in the clear. This is not acceptable for some specific systems, which transmit and receive extremely sensitive data. It is therefore critical that we somehow ensure that the data transmitted over the network is private (data privacy). FTP and Telnet, both are used a common way to remotely control Web servers; which is accomplished using Telnet, and corresponding services e.g., uploading a Web page file to a server which can easily be accomplished using FTP. Although FTP and Telnet both provide client connectivity, automation and end to end security but still they are not preferred in some specific cases because they don't provide data confidentiality, data integrity, authentication, access control; even the centralized control and cross platform support. So, it is quite obvious that the limitations of FTP and Telnet, simply make a better technology to replace both the ftp and telnet. SSH has evolved to be the obvious solution to the scenario because on the one hand, it has removed all the limitations of ftp and telnet, at the other hand, it incorporated some of its own certain advantages. This fact, we will discuss further in this

paper. The main purpose of SSH is to securely transmit data over network connections using strong encryption and authentication methods such as AES or triple DES. It includes, in commercial versions, high levels of technical support, maintenance, and confiscates network vulnerabilities. Standard methods are provided for setting up secure interactive shell sessions and for forwarding ("tunneling") arbitrary TCP/IP ports and port connections.

This paper is organized as follows. In section 2, we discuss the concept of FTP and Telnet. Section 3 discusses the SSH architecture. SSH features are discussed in Section 4. Section 5 describes the SSH versions including the SSH Tectia and its architecture. Section 6 presents a clear view about the attacks and security aspects of SSH. Section 8 concludes the paper.

II. THE CONCEPT OF FTP AND TELNET

FTP, short for File Transfer Protocol, the protocol for exchanging files over the Internet. FTP works in the same way as HTTP for transferring Web pages from a server to a user's browser and SMTP for transferring electronic mail across the Internet in that, like these technologies, FTP uses the Internet's TCP/IP protocols to enable data transfer. FTP is most commonly used to download a file from a server using the Internet or to upload a file to a server (e.g., uploading a Web page file to a server). Telnet, is basically a terminal emulation program for TCP/IP networks such as the Internet. The Telnet program runs on your computer and connects your PC to a server on the network. You can then enter commands through the Telnet program and they will be executed as if you were entering them directly on the server console. This enables you to control the server and communicate with other servers on the network. To start a Telnet session, you must log in to a server by entering a valid username and password. Telnet is a common way to remotely control Web servers.

FTP and telnet has following imperfections.

- FTP has one inherent problem: it is dangerously nonsecure. Any information transmitted via FTP travels over the network in clear text, which means anyone with a sniffer can read it. The logons to FTP servers require a user name and password, which also travel in unprotected clear text and can be easily grabbed by malicious individuals with just a hint of technical sense. This lack of security is particularly risky for

organizations governed by stringent regulations aimed at protecting sensitive data.

Keeping in mind, the inherent disadvantages of FTP & Telnet, it had become inevitable that a better technology has to break out. SSH is a replacement for other no secure protocol such as Telnet, FTP, X11, and Berkeley r-commands (rlogin, rcp, and rsh).

III. SSH ARCHITECTURE [1]

We can describe the SSH architecture in more enlighten way. Secure Shell (SSH) architecture has divided into three major components:

A. *The Transport Layer Protocol*

It provides server authentication, confidentiality, and integrity. It may optionally also provide compression. The transport layer will typically be run over a TCP/IP connection, but might also be used on top of any other reliable data stream. It provides server authentication, strong encryption and integrity protection

B. *The User Authentication Protocol*

It authenticates the Client-side user to the server. It runs over the transport layer protocol. It is used to authenticate the user on the client side to the server. The client sends a service request once a secure transport layer connection has been established. A second service request is sent after user authentication is complete.

C. *The Connection Protocol*

It multiplexes the encrypted tunnel into several logical channels. It runs over the user authentication protocol. It multiplexes the encryption tunnel into several logic channels. The connection protocol provides channels that can be used for a wide range of purposes.

IV. FEATURES OF SSH

The ten essential features for an effective network secure and successful protocol for an enterpriser user are highlighted below and provided by SSH:

A. *Industry-standard client connectivity*

We want a browser connection using an on-demand thin client. In others, you will need a command-line client that can be scripted for automation. In this scheme we can opt for a Windows client with a graphical user interface that makes file transfers easy for users.

B. *Data confidentiality*

The Internet is a public network, so it is imperative that you scramble sensitive data to make it unreadable during transmission. To that end, you need a standard means of encryption, such as Secure Sockets Layer (SSL), Transport Layer Security (TLS, the new version of SSL), or Secure Shell (SSH). These protocols all include a range of standard encryption ciphers that provide different strengths of encryption.

C. *Data integrity*

People with malicious intent thrive on tampering with data—for example, changing account balances or purchase order amounts. To ensure that bits remain intact as they travel from client to server or between servers, you will need some type of message authentication code. This code will also serve as a way to ensure message authenticity.

In addition, you will want an automatic way to restart an interrupted transfer at the point where the connection was dropped. A message authentication code can be provided by the secure shell technology.

D. *Authentication*

To launch a secure session, users must be able to verify their identities and establish their rights to connect to a given server. They must also be assured that they are connecting to the right server. SSH provides two-way authentication should take place in a secure way—either via strong authentication or via user name and password encryption without a solid authentication method, intruders can easily gain access to private files and systems via sniffing, social engineering, or password guessing.

E. *Automation*

Organizations often automate routine file transfers to save time and money. For example, large commercial businesses use batch processing extensively for back-office tasks.

Multiple files can be transferred in a batch when the load on back-office systems is light. These jobs are scheduled and typically run via scripts. Job scheduling are also provided by this. The key is to make sure that the scripts are using secure mechanisms to upload data.

F. *Access control*

Most organizations want to give users access only to specific directories on the internal network, which requires defining users and their access rights. Organizations also want to limit the number of open ports in their firewalls to minimize the chances of malicious traffic getting through. SSH give access only to authenticate users.

G. *Auditing*

If a client or server is compromised and data are retrieved illicitly, you will want to know what happened. By logging incoming and outgoing transmissions, auditing tools can help identify suspicious usage patterns, inadvertent or overlooked access, and mischievous or purposeful attacks. These tools are crucial to discovering both apparent and subtle forms of intrusion, making them a critical part of your security solution.

H. *Centralized management*

In order to meet growing demands for information from whole new audiences of users, centralized management of the client side of the file-transfer process is essential. In fact, it is the only effective way to tackle today's fast changing configuration and deployment requirements.

I. Cross-platform support

Your customers or partners may be running on platforms that are different from your own, which is probably heterogeneous. For that reason, the ideal secure file transfer solution will provide cross-platform support.

J. End-to-end security

To move a file securely across the Internet is one thing; getting that file securely to the end point (the server that it updates) is another. The vast majority of financial fraud is the result of internal hacking rather than external penetration of a secure network, so the last leg of protection, from the DMZ to the back office, is critical. To guard against these inside jobs, you need end-to-end security.

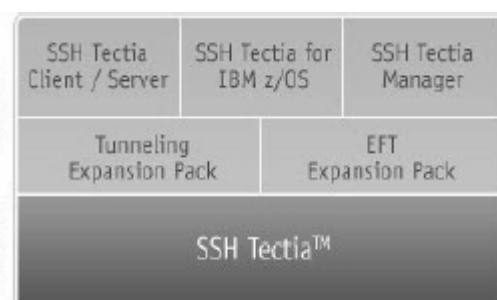
V. SSH Versions

Tata Ylönen developed SSH1 in 1995. His initial impulsion was to address the no secure data—passwords, e-mails, protocol interfaces. First version SSH1 features for Client programs is that perform remote logins, remote command execution, and secure file copying across a network. It also provide an extremely configurable SSH server. It uses several selectable encryption algorithms and authentication mechanisms. It has an SSH agent to cache keys for easy access. But there were several limitations in the protocol. Then Ylönen developed second version of SSH i.e. SSH2 in 1996. SSH2 features it also includes the above but also include some stronger ones. It has Encryption ciphers, such as 3DES and AES. It can use of sound cryptographic Message Authentication Code (MAC) algorithms for integrity checking. It has major Support for public key certificates. SSH1 and SSH2 are based on different protocols. A free version for home users is also available i.e. Open SSH. It includes a large user base, support for most UNIX platforms, But Open SSH comes with significant risks that administrators must consider when choosing an SSH provider. The Platform selection is limited and lacks native Windows support. There is little or no quality control. Several critical updates need to be installed every year, and even that is risky. There is no single point of reference for updates. There is no reliable support, no responsible party to go to with questions, and always the possibility of a unfriendly reply in response to the simplest of newsgroup queries. Functionality it is technically limited. For example, PKI lacks user authentication, which is available with supported SSH products. Open SSH is behind the curve by more than major product enhancements and will not catch up with the supported technology. There are no liability or patent guarantees, leaving companies vulnerable to lawsuits over copyrights and patents. The next is Reflection for Secure IT a new version includes High-quality software and reduced administrative overhead and low total cost of ownership. It also provides Full support and maintenance, including a professional staff to answer questions, provide custom design, and solve problems. It has Timely software updates and the opportunity to do business in new ways.

A. The SSH Tectia Approach

It is a new commercial version. Which removes all above attacks? The high performance SSHG3 architecture can provide up to 150% improvement in cryptographic performance compared to other secure solutions. SSH Tectia is the leading end-to-end communications security solution from the original developers of Secure Shell. SSH Tectia allows secure system administration and file transfer in cross-platform enterprise environments consisting of Windows, Unix, Linux, and mainframes. With the SSH Tectia solution, organizations can easily replace unsecured FTP, Telnet, and Unix r-series commands with standards-based Secure Shell technology. It has special features as Cross-platform support that provides tested and supported binary solutions for Windows, Unix, Linux, and mainframes allows standardization on a single Secure Shell solution. Centralized deployment, configuration, and maintenance reduce total costs. Centralized monitoring improves security and facilitates regulatory compliance. Broad authentication support allows easy integration into enterprise environments. The high-performance SSH G3™ implementation of the standard Secure Shell offers higher throughput and scalability. Reliable support services from the original developers of Secure Shell ensure easy deployment and maintenance of SSH Tectia. SSH Tectia is the leading Secure-Shell-based end-to-end communications security solution for large enterprises, financial institutions, and government agencies. SSH Tectia provides secure system administration, secure file transfers (SFTP and SCP), and secure application connectivity (Secure Shell tunneling). The centralized management capabilities of SSH Tectia Manager support centralized deployment, maintenance, and monitoring of communications security, facilitating improved regulatory compliance and reduced total costs. The SSH Tectia solution is available for a broad range of platforms including common Unix, Linux, Windows, and IBM mainframe operating systems. The goal was to increase the protocol performance to a new level so that Secure Shell would not become a processing bottleneck even in the most demanding environments.

SSH Tectia Architecture



An new optimized architecture which has improved protocol throughput, Enhanced connection scalability with highly optimized memory consumption, Improved encryption and

data authentication performance with the use of the CryptiCore algorithm .Connection Broker One of the biggest architecture-level changes in SSH G3 is the introduction of a component called the Connection Broker. The Connection Broker is a common component for all client-side Secure Shell programs. The main benefit of handling all client-side connections from a single point is to avoid replication of code and policies resulting in ease of use and reduced memory consumption. For example, all clients, including Windows clients with GUI (SSH Tectia Client and Connector), and command-line tools can share the same configuration file and settings. Security is also further improved by isolating all security-critical operations including authentication data handling in a single component. Additionally, the Connection Broker enables easy integration of Secure Shell functionality to client applications without the need to integrate the whole protocol stack to each application.

As improved throughput is one of the main objectives for the new protocol implementation, special attention was given to code paths that involve processing the Secure Shell channel data and the actual payload. For example, the number of data copy operations in that code has been reduced to absolute minimum to minimize the throughput time. The new Secure Shell server architecture in SSH G3 is also multi-threaded making it possible to fully leverage multi-processor servers for better performance. SSH G3 implements n x m server process architecture for optimized server-side memory consumption and performance. Most other Secure Shell server implementations always create a new server-side process when a new connection is established to the server. For example, in secure application connectivity (tunneling) this leads to a large number of server-processes, each used for running a single tunnel. As a consequence, the memory usage becomes inefficient. In the new n x m server process architecture there are always n + 1 server processes running: one master server distributes the connections to n servant servers. While each servant server process can handle a maximum of m concurrent connections, the maximum supported number of concurrent connections becomes n x m. Depending on the environment and tunneled applications, this number can in practice be thousands. It concludes it has server side scalability.

B. CryptiCore Algorithm

SSH G3 architecture supports a broad range of encryption ciphers including 3DES and AES. Additionally, the algorithm support has now been expanded to include CryptiCore@ encryption for an extra boost in encryption performance on Intel-based platforms. The CryptiCore encryption and data authentication technology, developed by a Danish data security company *Cryptico*, is software-based and fully processor optimized to offer a significantly higher speed than other encryption and data authentication software. The CryptiCore encryption is based on the Rabbit stream cipher, which has been presented at leading international conferences and endorsed by experts in the field.

While all performance-specific Secure Shell protocol improvements can be experienced with all supported ciphers including the FIPS 140-2 crypto, using CryptiCore brings further throughput improvements and is well suited for data-intensive environments (see the next chapter for quantitative performance data).

Compared product	Throughput Mbit/s	Increase compared to slowest
OpenSSH 4.1	210	-
SSH Tectia client/server solution 5.0	250	+ 19%

VI. SECURITY ASPECTS OF SSH

Several vulnerabilities and attacks lie in the path when communication done, some attacks which are common and confiscated by SSH some are described as following as:

A. Eavesdropping

By using packet-sniffer software, eavesdroppers can read unencrypted network traffic without the sender or receiver knowing it. Sniffer can collect unencrypted passwords, Usernames and any other data transferred via the network. Even if a password is encrypted, eavesdroppers can capture personal information simply by watching the unencrypted traffic that's generated after the password is entered. SSH can protect against eavesdropping by encrypting all data, making it unreadable to potential eavesdroppers.

B. DNS and IP spoofing

Domain Name System spoofing is what happens when an attacker traps a DNS server. This allows someone to access your site's e-mail or direct users to the wrong web pages. IP spoofing is a technique that grants access to a computer by sending a message with the IP address of a trusted host .This takes a little work on the attacker's end, but it is an effective hacking method. SSH wards off such attacks by cryptographically verifying the identity of the server, for every session, the SSH client validates the server's host key against a local list of available keys that are associated with server names and addresses. If the keys do not match, then an immediate warning is issued.

C. Connection hijacking

When an attacker disconnects users from their TCP connections, it's called connection hijacking. An active attacker can listen to network traffic as well as inject their own traffic into the transmission—in order to perform a connection hijacking. When hijacking a connection, an attacker sits on the network, sniffing for packets transmitted from the client to the server. The attacker then obtains the hosts' IP addresses and relative port numbers, which allow him to spoof TCP/IP packets. This kind of attack can be devastating, regardless of how strong the authentication method is. SSH is unable to

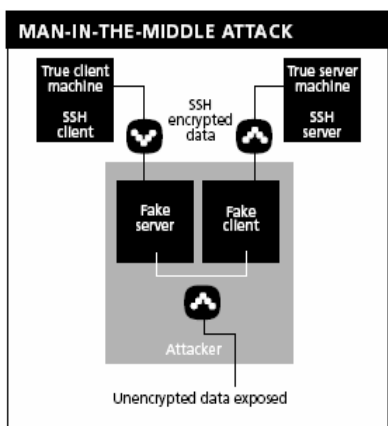
prevent hijackings because of an inherent flaw in the TCP layer. SSH can, however, render the hijacking ineffective through its integrity-checking process. If a session is modified during transmission, SSH will shut down the connection immediately—without using the nefarious data. SSH2 uses the cryptographically strong hash functions MD5 and SHA-1 for integrity checking.

D. Man-in-the-middle attacks

A man-in-the-middle attack occurs when a third party poses as both ends of a communication. During a session, an attacker will sit between the SSH client and the server and convince each of the two hosts that he is the other host. After a client has been authenticated and given access by a server, the attacker can learn which port and sequence numbers are being used to transmit data. With this information, the attacker can intercept traffic and read, capture, or delete data. The hacker shares the session key

With the legitimate user, fooling both the client and the server into thinking they are connected to one another. There are two ways SSH can protect against man-in-the middle attacks: The first is SSH's server-host authentication. Because the attacker does not have the server's private host key, he would have to break into the server host to pull off the imitation. For this effective protection, the client must check the server-supplied public host key against its list of known hosts. The second defense SSH provides is stronger authentication for the client. Passwords are vulnerable, but public keys and certificates are essentially immune to these types of attacks. The following diagram

Shows that how SSH detect and removes man in middle attack.



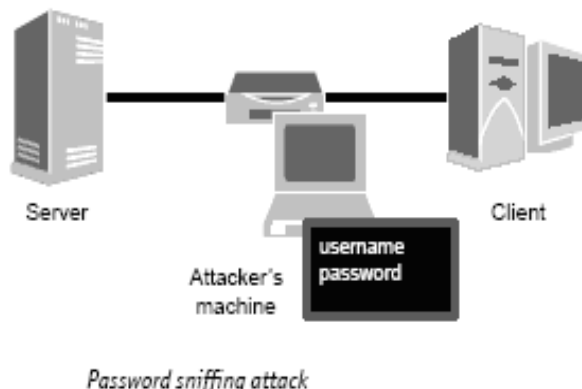
E. Insertion attacks

It also known as a replay attack, a hacker identifies a packet containing a specific command and then resends the packet at another time during the session in order to replay the command. Like TCP connection hijacking, insertion attacks introduce altered data into the network stream. But in the case of insertion attacks, the insertion is encrypted and sent to either the client or server to decrypt. SSH2 uses

cryptographically strong integrity checking namely SHA-1 and MD5, to repel insertion attacks.

F. Password Exposure

The user authentication mechanisms of both FTP and Telnet rely on passwords that are transmitted in plaintext over the network. Anyone with network access can easily collect the passwords using standard network monitoring or dedicated sniffing tools that are easily available. Once the password has been exposed, the attacker can gain access to managed servers. This diagram shows that how it counters.



SSH provides a strong encrypted tunnel which does not give the pass to this to enter into network.

G. Attacks which are not cracked by SSH1, SSH 2.

SSH has no defense once an attacker has gained root access to a machine. At that point, the attacker can destabilize SSH from within and render security nonexistent. The following examples illustrate the types of attacks

Password cracking

Although SSH encrypts passwords while they're transmitted over the network, a password is an authentication type that's easily compromised the obvious counter to this kind of attack is to come up with a difficult password. With SSH, servers can also be configured to disallow the use of passwords; public keys can be used.

Traffic analysis

SSH traffic is easy to spot and is customarily destined to well-known place. The amount of traffic, the destination, and the timing can all be used by a third party to identify transactions, backup schedules, or the best time of day to launch a denial-of-service attack. One way to mitigate this risk is to produce a constant stream of traffic, whether the network is active or not. This prevents prying eyes from being able to pick out the true traffic.

Covert channels

SSH does nothing to counter covert channels. Covert channels transmit data in an unanticipated and unnoticed manner. For example, employees whose e-mail use is restricted might

communicate with one another by including messages in their home directories. A system administrator wouldn't expect this sort of communication and would have a difficult time eliminating it. It can, however, be used as a covert channel itself—with even/odd packet lengths simulating the dots and dashes of Morse code

VII. LIMITATIONS OF SSH

A. Cross Platform Connectivity

Few organizations can commit to a single platform environment on both workstation and server side. Most large enterprise networks consist of Windows, Unix, Linux, and mainframe systems. One of the benefits of FTP is its multi-platform support that allows transferring files between servers in heterogeneous environments. Telnet, on the other hand, allows effortless terminal connections from Windows workstations to Unix and Linux systems. When replacing FTP and Telnet with secure alternatives, it is important to maintain the option for smooth cross-platform connectivity..

B. User Experience

FTP and Telnet have become real tools for many enterprise users, who are involved in setting up and maintaining file transfers. Changing their user experience and introducing a completely different kind of process of transferring files will require user training and increase the likelihood of errors before the users become familiar to the new tools.

C. Monitor and Compliance

It is not enough to ensure that all file transfers and system administration connections are strongly encrypted and authenticated. The ability to monitor secured access is needed as well for effective security management.

D. Authentication Integration

Perhaps the most severe vulnerability in FTP and Telnet is the fact that passwords are transmitted in plaintext over the network. Password sniffing can be eliminated with the use of hashing, as implemented by most security solutions. But passwords can still easily lead to weak authentication since users tend to choose inherently weak passwords, or too easily expose their passwords to others. Strong user authentication methods include hardware authenticators that produce one-time passwords (e.g. RSA SecurID) and PKI-based smart cards and USB tokens that implement cryptographic operations inside the device.

E. Cryptographic Performance

When implementing data encryption, the additional processing required for cryptographic operations . it always introduces a penalty to performance. While this is unavoidable, there are different ways how security products can keep their impact on CPU and memory usage, data throughput, and system scalability at a minimum. Differences in supported algorithms and their implementations can result

in significant differences in performance of otherwise similar products that support the same security protocols. Since performance results are highly dependent on the test environment where they are produced, it is important to simulate the real environment during testing. A performance penalty can be particularly costly in mainframe environments, where system costs are commonly based on the CPU usage. The option to seamlessly use dedicated hardware devices for cryptographic operations is a necessity in mainframe security products and can yield major cost savings.

CONCLUSION

An analytical approach for finding out the most cost-effective security solutions that optimize the usage of key IT resources within constrained budgets. The description provides exactly that approach. Intangible maintenance costs consisting of resource-consuming administration work often constitute the largest part of the total costs. Specifically, FTP/Telnet security projects can involve many highly time-consuming administrative areas such as compiling, testing, installing, updating, configuring, managing authentication keys etc. By choosing solutions that can reduce or completely eliminate many of these cost items, organizations can achieve major cost-savings. It can reduce the software management, reduce internal support costs, and simplify vendor relationship coordination resulting in reduced total costs. Analysis can also be applied for evaluating possibilities to save costs in existing environments where FTP, Telnet, and other enterprise applications have already been secured. In many cases the cost of acquiring and migrating to a new solution can have lower total costs than holding to an existing expensive-to-maintain security system, thus creating a positive return on investment.

FUTURE WORK

The worst problem, however, is that SSH is not very compatible with network-based intrusion detection. In the last few years attackers have been using SSH to connect from one system to another, particularly in connection with powerful Root kit tools such as SucKIT that they install in compromised solutions. But there are some problems one problem, for example, is that users (and sometimes even system administrators) who use SSH often tend to become complacent about security. They assume that their computing environment is secure simply because they use this protocol. They may consequently neglect other critical considerations, such as configuring their systems properly and patching vulnerabilities. Open SSH has been plagued by security flaws such as buffer management errors, PAM (Pluggable Authentication Module) challenge authentication failures, PAM conversion stack corruption, timing problems that can allow attackers to determine RSA keys, and others that can result in a range of undesirable outcomes, including execution of rogue code, encryption key compromise, and denial of service. The fact that SSH-based attacks have suddenly become so common and that powerful tools such as SucKIT are now so widely available, makes careful analysis of the cost to benefit equation for SSH even more very important.

REFERENCES

- [1]. Ylonen ,T. and Lonvick, C., “The Secure Shell (SSH) Protocol Architecture”, Ed. Cisco Systems, Inc., January 2006.
- [2]. Rosasco, Nicholoas, larochele David, “How and why more secure technologies succeed in legacy markets lessons from the success of SSH”
- [3]. Barrett, Daniel J. and Silverman, Richard E.: “SSH the Secure Shell”, O'REILLY, February 2001.
- [4]. Bertrand, Louis., “How SSH was freed”, Daemon News (Dec. 1999)
- [5]. Adams., Carlisle. and Zuccherato., Robert. “A General, Flexible Approach to Certificate Revocation,” Entrust technologies, White Paper, June 1998.
- [6].<http://www.ssh.com/>
- [7]. <http://openssh.com/index.html>
- [8]. Ylonen. T “The SSH (Secure Shell) Remote login Protocol”, Internet Draft, Networking Group, November 1995.
- [9]. Shuo Chen., Xu., Zbigniew, and Iyer, Ravishankar”, An Experimental Study of Security Vulnerabilities Caused by Errors”.
- [10]. Using SSH : Do Security risks Outweigh the benefits? A white paper by Gene Schultz.