CSCI-GA.3205 Applied Cryptography & Network Security

Department of Computer Science
New York University

PRESENTED BY DR. MAZDAK ZAMANI
mazdak.zamani@NYU.edu

Administering a Secure Network
TCP/IP Protocols
Monitoring and analyzing logs
Port security
HTTPS, SSH, TLS/SSL

10

NYU

CompTIA Security+ Guide to Network Security Fundamentals, Fifth Edition

Chapter 8
Administering a Secure Network

Network Administration Principles

- Administering a secure network can be challenging. Rule-based management approach relies on following procedures and rules.
- Procedural rules dictate technical rules. Technical rules address:
    - Device security
    - Monitoring and analyzing logs
    - Network design management
    - Port security

Device Security

Task	Explanation
Create a network design	Prior to any configuration, a network diagram that illustrates the router interfaces should be created. This diagram should reflect both the LAN and wide area network (WAN) interfaces.
Use a meaningful router name	Because the name of the router appears in the command line during router configuration, it helps ensure that commands are given to the correct router. For example, if the name *Internet_Router* is assigned to the device, the displayed command prompt would be *Internet_Router (config)#*.
Secure all ports	All ports to the router should be secured. This includes both physical ports (sometimes called the *console port* and *auxiliary port*) and inbound ports from remote locations (sometimes known as *VTY* for *virtual teletype*).
Set a strong administrator password	Most routers allow a user to access the command line in *user mode*, yet an administrator password is required to move to *privileged mode* for issuing configuration commands.
Make changes from the console	The configuration of the router should be performed from the console and not a remote location. This configuration can then be stored on a secure network drive as a backup and not on a laptop or USB flash drive.

Table 8-4 Secure router configuration tasks

Monitoring and Analyzing Logs

Device	Explanation
Firewalls	Firewall logs can be used to determine whether new IP addresses are attempting to probe the network and if stronger firewall rules are necessary to block them. Outgoing connections, incoming connections, denied traffic, and permitted traffic should all be recorded.
Network intrusion detection systems (NIDS) and network intrusion prevention systems (NIPS)	Intrusion detection and intrusion prevention systems record detailed security log information on suspicious behavior as well as any attacks that are detected. In addition, these logs also record any actions NIPS used to stop the attacks.
Web servers	Web servers are usually the primary target of attackers. These logs can provide valuable information about the type of attack that can help in configuring good security on the server.
DHCP servers	DHCP server logs can identify new systems that mysteriously appear and then disappear as part of the network. They can also show what hardware device had which IP address at a specific time.
VPN concentrators	VPN logs can be monitored for attempted unauthorized access to the network.
Proxies	As intermediate hosts through which websites are accessed, these devices keep a log of all URLs that are accessed through them. This information can be useful when determining if a zombie is "calling home."
Domain Name System (DNS)	A DNS log can create entries in a log for all queries that are received. Some DNS servers also can create logs for error and alert messages.
Email servers	Email servers can show the latest malware attacks that are being launched through the use of attachments.
Routers and switches	Router and switch logs provide general information about network traffic.

Table 8-5 Device logs with beneficial security data

NYU

Network Design Management

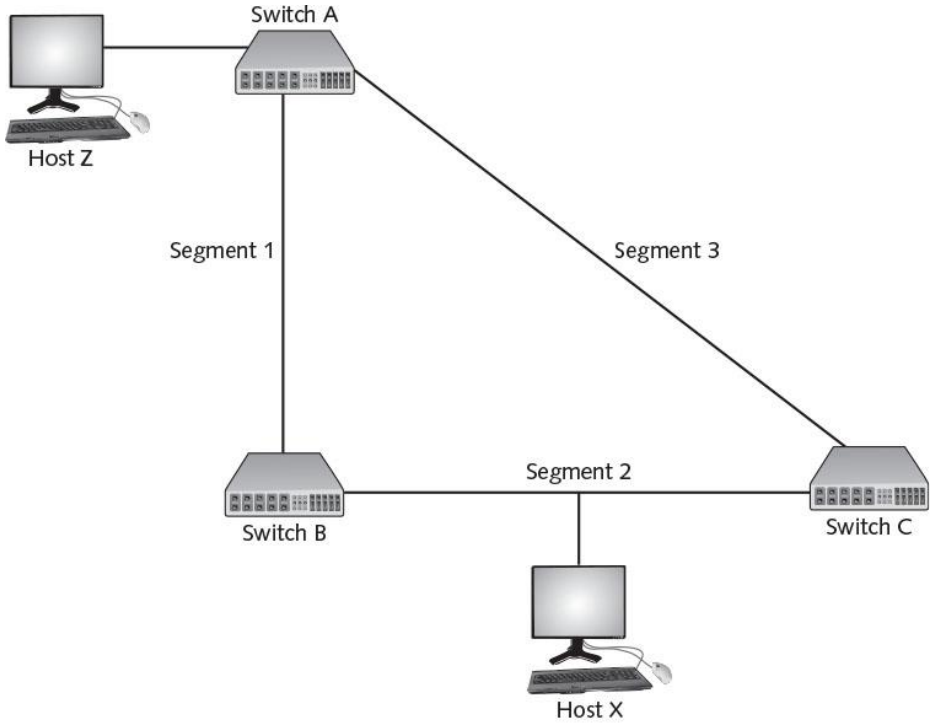


Figure 8-7 Broadcast storm

Port Security

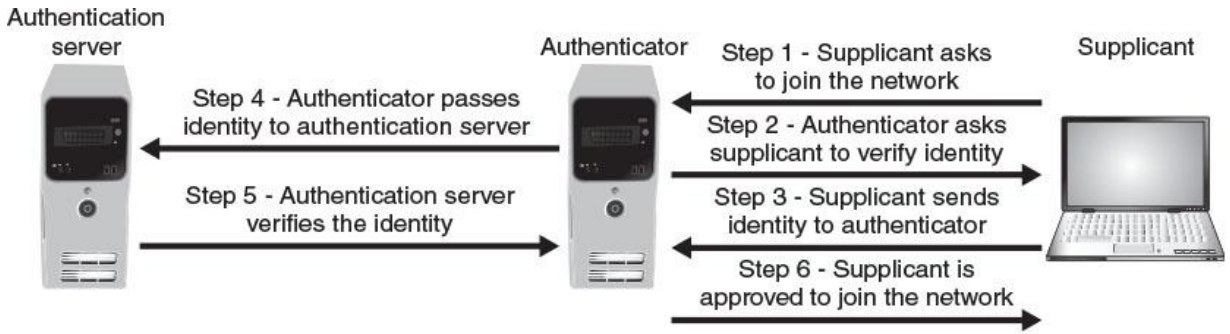


Figure 8-8 IEEE 802.1x process

Common Network Protocols

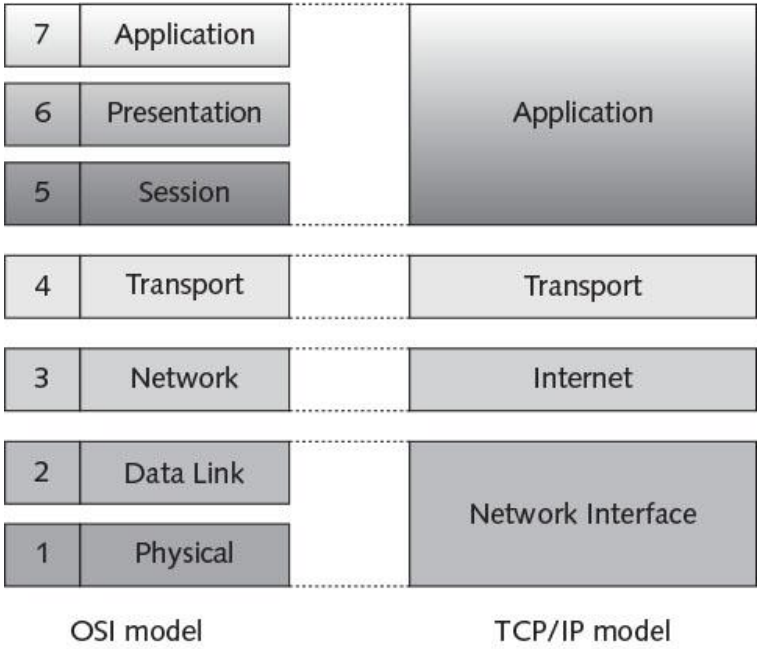


Figure 8-1 OSI model vs. TCP/IP model

Common Network Protocols

- Protocols
  - Rules for communication
  - Essential for proper communication between network devices
- Transmission Control Protocol/Internet Protocol (TCP/IP)
  - Most common protocol suite used for local area networks and the Internet
  - Comprises several protocols that all function together

Common Network Protocols

- IP
  - Protocol that functions primarily at Open Systems Interconnection (OSI) Network Layer (Layer 3)
  - Provides network addressing and routing
- TCP
  - Transport Layer (Layer 4) protocol
  - Establishes connections and ensures reliable data transport between devices
- TCP/IP uses a four-layer architecture
  - Network Interface, Internet, Transport, Application

Common Network Protocols

- Several basic TCP/IP Protocols:
  - Internet Control Message Protocol (ICMP)
  - Simple Network Management Protocol (SNMP)
  - Domain Name System (DNS)
  - File transfer and storage protocols

Common Network Protocols

TCP/IP Layer	Protocols	Security Protocol
Application	DNS, FTP, SNMP, HTTP	PGP, HTTPS
Transport	TCP, UDP	SSL, TLS, SSH
Internet	IPv4, IPv6, ARP, ICMP	IPsec
Network Interface	IEEE 802.2	WEP, WPA2, WPS, 802.1X

Internet Control Message Protocol (ICMP)

Used by devices to communicate updates or error information to other devices. ICMP messages are divided into two classes:

- Informational and query messages
- Error messages

- ICMP message fields
  - Type: Identifies general message category
  - Code: Gives additional information about the Type field
  - Checksum: Verifies message integrity
  - Message Body: Contains information about the specific ICMP message

Internet Control Message Protocol (ICMP)

- Attacks that take advantage of ICMP
  - Network discovery
  - Smurf attack
  - ICMP redirect attack
  - Ping of death

Type 3 code value	Description
0	Destination network unreachable
1	Destination host unreachable
2	Destination protocol unreachable
3	Destination port unreachable
5	Source route failed
6	Destination network unknown
7	Destination host unknown
9	Communication with destination network administratively prohibited
12	Host unreachable for Type of Service

Table 8-1 Common ICMP code values for Type 3, Destination Unreachable

NYU

Simple Network Management Protocol (SNMP)

- Used to manage network equipment and is supported by most network equipment manufacturers
- Allows administrators to remotely monitor, manage, and configure network devices
- Functions by exchanging management information between network devices
- Each SNMP-managed device has an agent or a service
  - Listens for and executes commands
- Agents are password protected
  - Password is known as a *community string*
- Security vulnerabilities were present in SMNP versions 1 and 2
  - Version 3 uses usernames and passwords along with encryption to address vulnerabilities

Domain Name System (DNS)

- A TCP/IP protocol that maps IP addresses to their symbolic name
- The DNS database is organized as a hierarchy
  - Database consists of the name of a site and a corresponding IP number
- DNS is often the focus of attacks
  - DNS poisoning substitutes fraudulent IP address

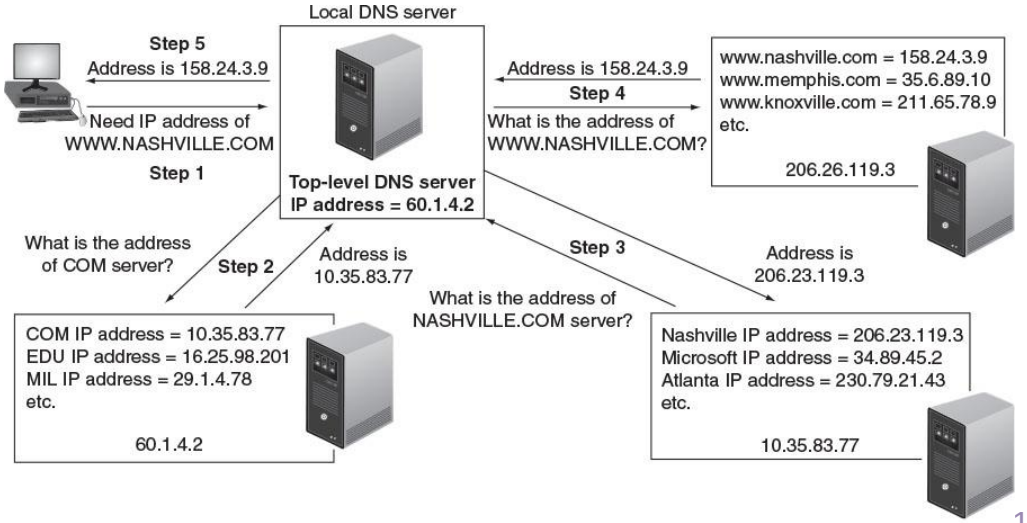


Figure 8-2 DNS lookup

File Transfer Protocols

- File transfer protocol (FTP) - used to connect to an FTP server. FTP uses two ports:
  - Port 21 is the FTP control port
  - Port 20 is the data port
- FTP vulnerabilities
  - Does not use encryption
  - Files transferred using FTP are vulnerable to man-in-the-middle attacks

- Secure sockets layer (FTPS) encrypts commands
  - Uses SSL or TLS to encrypt commands sent over the control port (port 21); data port may not be encrypted
- Secure FTP (SFTP)
  - Uses only a single TCP port instead of two ports
  - All data and commands are encrypted

HTTPS
SSH
TLS/SSL

SSL/TLS

- Secure Sockets Layer (SSL) and its next substitute, Transport Layer Security (TLS), are encryption protocols designed to ensure the security, reliability, and authenticity of the information exchanged.
- Although SSL is replaced by TLS, the industry uses the two terms interchangeably. It consists of two parts:
  - a **handshake protocol** that performs authenticated key exchange to establish the shared keys, and
  - a **record-layer protocol** that uses those shared keys to encrypt/authenticate the parties' communication.

SSL/TLS - The handshake protocol

The handshake protocol. We now describe the basic flow of the handshake protocol. At the outset of the protocol, the client C holds a set of CA's public keys $\{pk_1, \ldots, pk_n\}$, and the server S has a key-pair (pk_S, sk_S) for a KEM[9] along with a certificate $\mathsf{cert}_{i \to S}$ issued by one of the CAs whose public key C knows. To connect to S, the parties run the following steps.

1. C begins by sending a message to S that includes information about versions of the protocol supported by the client, the *ciphersuites* supported by the client (e.g., which hash functions or block ciphers the client allows), and a uniform value (a "nonce") N_C.

2. S responds by selecting the latest version of the protocol it supports as well as an appropriate ciphersuite. In addition, it sends its public key pk_S, its certificate $\mathsf{cert}_{i \to S}$, and its own uniform value N_S.

SSL/TLS - The handshake protocol

3. C checks whether one of the CA's public keys it holds, say pk_i, matches the CA who issued S's certificate. If so, C verifies the certificate (and also checks that it has not expired or been revoked) and, if successful, learns that pk_S is S's public key. It then runs $(c, \mathsf{pmk}) \leftarrow \mathsf{Encaps}_{pk_S}(1^n)$ (see Section 11.3) to obtain a ciphertext c and what is called a *pre-master key* pmk. It sends c to the server.

   pmk is used to derive a *master key* mk using a key-derivation function (cf. Section 5.6.4) applied to pmk, N_C, and N_S. The client then applies a pseudorandom generator to mk to derive four keys k_C, k'_C, k_S, k'_S.

   Finally, C computes $\tau_C \leftarrow \mathsf{Mac}_{\mathsf{mk}}(\mathsf{transcript})$, where transcript denotes all messages exchanged between C and S thus far. The client then sends τ_C to S. (In fact, τ_C is itself encrypted and authenticated as done for communication in the record-layer, described below.)

SSL/TLS - The handshake protocol

4. S computes $\mathsf{pmk} := \mathsf{Decaps}_{sk_S}(c)$, from which it can derive mk and k_C, k'_C, k_S, k'_S as the client did. If $\mathsf{Vrfy}_{\mathsf{mk}}(\mathsf{transcript}, \tau_C) \neq 1$, then S aborts. Otherwise, it sets $\tau_S \leftarrow \mathsf{Mac}_{\mathsf{mk}}(\mathsf{transcript}')$, where $\mathsf{transcript}'$ denotes all messages exchanged between C and S thus far (i.e., including the last message received from C). S then sends τ_S to C. (Again, τ_S is actually encrypted and authenticated as record-layer traffic is.)

5. If $\mathsf{Vrfy}_{\mathsf{mk}}(\mathsf{transcript}', \tau_S) \neq 1$, the client aborts.

At the end of a successful execution of the handshake protocol, C and S share a set of four symmetric keys k_C, k'_C, k_S, k'_S.

SSL/TLS - The record-layer protocol

The record-layer protocol. Once keys have been agreed upon by C and S, the parties use those keys to encrypt and authenticate all their subsequent communication. C uses k_C (resp., k'_C) to encrypt (resp., authenticate) all messages it sends to S; similarly, S uses k_S and k'_S to encrypt and authenticate all the messages it sends. Sequence numbers are used to prevent replay attacks, as discussed in Section 4.5.3. TLS 1.2 uses an authenticate-then-encrypt approach, which, as we have seen in Section 4.5.2, can be problematic.

TLS 1.3 record protocol

TLS 1.3 uses a nonce-based AEAD cipher (E, D) to encrypt a record. Which nonce-based AEAD cipher is used is determined by negotiation during TLS session setup. The AEAD encryption algorithm is given the following arguments:

- secret key: $k_{b \to s}$ or $k_{s \to b}$ depending on whether the browser or server is encrypting.
- plaintext data: up to 2^{14} bytes.
- associated data: empty (zero length).
- nonce (8 bytes or longer): the nonce is computed by (1) padding the encrypting party's 64-bit write sequence number on the left with zeroes to the expected nonce length and (2) XORing this padded sequence number with a random string (called `client_write_iv` or `server_write_iv`, depending on who is encrypting) that was derived from the master secret during session setup and is fixed for the life of the session. TLS 1.3 could have used an equivalent and slightly easier to comprehend method: choose the initial nonce value at random and then increment it sequentially for each record. The method used by TLS 1.3 is a little easier to implement.

TLS 1.3 record protocol

The AEAD cipher outputs a ciphertext c which is then formatted into an encrypted TLS record as follows:

`type`	`version`	`length`	ciphertext c

where `type` is a 1-byte record type (handshake record or application data record), `version` is a legacy 2-byte field that is always set to 0301, `length` is a 2-byte field indicating the length of c, and c is the ciphertext. The type, version, and length fields are all sent in the clear. Notice that the nonce is not part of the encrypted TLS record. The recipient computes the nonce by itself.

The length field

- In TLS 1.3, the record length is sent in the clear. Several attacks based on traffic analysis exploit record lengths to deduce information about the record contents.
- When encrypting a TLS record the length field is not part of the associated data and consequently has no integrity protection. The reason is that due to variable length padding, the length of c may not be known before the encryption algorithm terminates.
- Therefore, the length cannot be given as input to the encryption algorithm. This does not compromise security: a secure AEAD cipher will reject a ciphertext that is a result of tampering with the length field.

Replay prevention

- An attacker may attempt to replay a previous record to cause the wrong action at the recipient.
  - For example, the attacker could attempt to make the same purchase order be processed twice, by simply replaying the record containing the purchase order.
  - TLS uses the 64-bit write sequence number to reject such replicated packets. TLS assumes in-order record delivery so that the recipient already knows what sequence number to expect without any additional information in the record.
  - A replicated or out-of-order record will be discarded because the AEAD decryption algorithm will be given the wrong nonce as input causing it to reject the ciphertext.

The cookie cutter attack

- TLS provides a streaming interface, where records are sent as soon as they are ready. While replay, re-ordering, and mid-stream deletion of records is prevented by a 64-sequence number, there is no defence against deletion of the last record in a stream.
- In particular, an active attacker can close the network connection mid-way through a session, and to the participants this will look like the conversation ended normally.
- This can lead to a real-world attack called cookie cutter.

The cookie cutter attack - Example

- Consider a victim web site and a victim web browser. The victim browser visits a malicious web site that directs the browser to connect to victim.com. Say that the encrypted response from the victim site looks as follows:

```
HTTP/1.1 302 Redirect
Location: https://victim.com/path
Set-Cookie: SID=[AuthenticationToken]; secure
Content-Length: 0 \r\n\r\n
```

  - The first two lines indicate the type of response. Notice that the second line includes a "*path*" value that is copied from the browser's request.
  - The third line sets a cookie that will be stored on the browser.
  - Here the "*secure*" attribute indicates that this cookie should only be sent to victim.com over an encrypted TLS session.
  - The fourth line indicates the end of the response.

The cookie cutter attack

Suppose that in the original browser request, the “path” value is sufficiently long so that the server's response is split across two TLS frames:

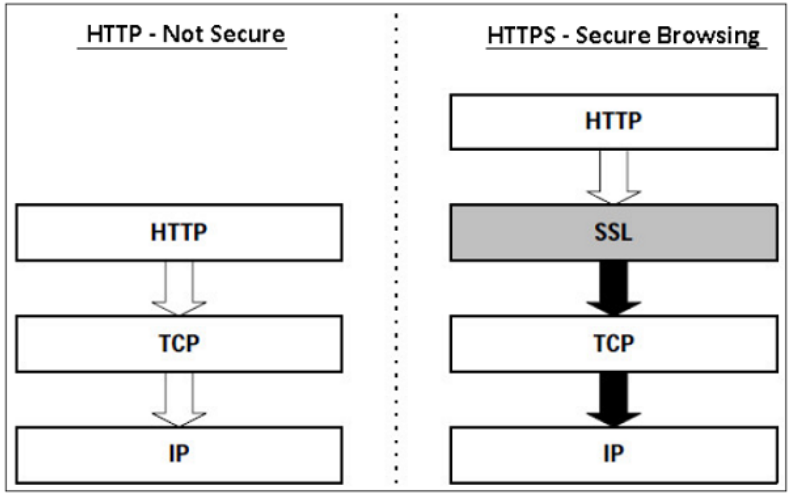
```
frame 1:    HTTP/1.1 302 Redirect
            Location: https://victim.com/path
            Set-Cookie: SID=[AuthenticationToken]

frame 2:    ; secure
            Content-Length: 0  \r\n\r\n
```

- The network attacker shuts down the connection after the first frame is sent, so that the second frame never reaches the browser.
- This causes the browser to mark the cookie as non-secure.
- Now the attacker directs the browser to the cleartext (http) version of victim.com, and the browser will send the SID cookie in the clear, where the attacker can easily read it.

HTTPS

- Hyper Text Transfer Protocol (HTTP) protocol is used for web browsing.
- The HTTPS (HTTP over SSL) provides “secure” web browsing.
- Secure Sockets Layer (SSL)/ Transport Layer Security (TLS) protocol is used to provide the encrypted and authenticated connection between the client web browser and the website server.

- The following content are encrypted:
  - URL of the requested web page.
  - Web page contents provided by the server to the user client.
  - Contents of forms filled in by user.
  - Cookies established in both directions.



The process of HTTPS

- You request a HTTPS connection to a webpage by entering https://...
- Web browser initiates a connection to the web server. (Use of SSL protocol)
- An application, browser in this case, uses the system port 443 instead of port 80.
- The SSL goes through a handshake protocol for establishing a secure session.
- The website initially sends its SSL Digital certificate to your browser. On verification of certificate, the SSL handshake progresses to exchange the shared secrets for the session.
- When a trusted SSL Digital Certificate is used by the server, users get to see a padlock icon in the browser address bar. When an Extended Validation Certificate is installed on a website, the address bar turns green.
- Once established, this session consists of many secure connections between the web server and the browser.

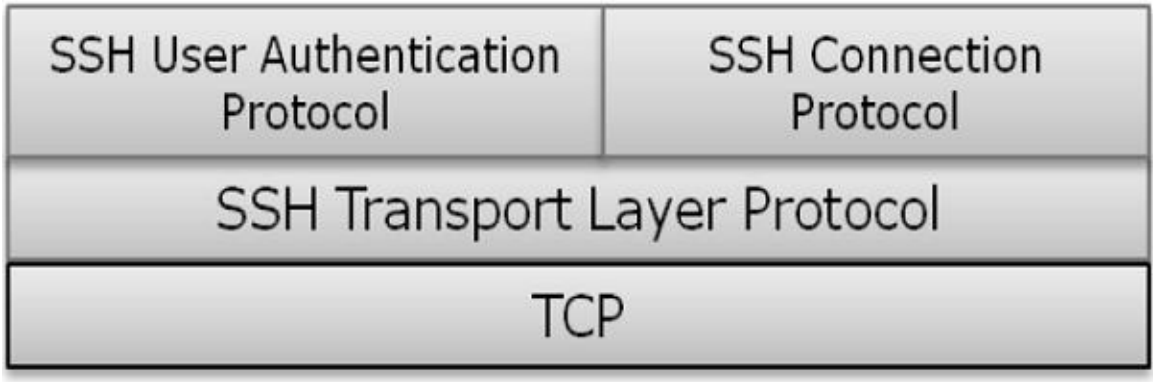
Secure Shell Protocol (SSH)

- Telnet is used to communicate with a remote server, and it does not use any security mechanism and transfers the data over network/internet in a plain-text.
- SSH, also known as Secure Shell or Secure Socket Shell, is a network protocol that provides administrators with a secure way to access a remote computer.
- SSH establishes a cryptographically secured connection between two parties (client and server), authenticating each side to the other, and passing commands and output back and forth.

Secure Shell Protocol (SSH)

- SSH protocol uses **symmetric encryption, asymmetric encryption and hashing** in order to secure transmission of information.
- The SSH connection between the client and the server happens in three stages:
  - Verification of the server by the client.
  - Generation of a session key to encrypt all the communication.
  - Authentication of the client.

Three sub-protocols of SSH



Three sub-protocols of SSH - Connection Protocol

- Connection Protocol – This provides multiple logical channels over a single underlying SSH connection.

Three sub-protocols of SSH - Transport Layer Protocol

- Transport Layer Protocol – This part of SSH protocol provides data confidentiality, server (host) authentication, and data integrity. It may optionally provide data compression as well.

- Transport Layer Protocol
  - Server Authentication
  - Session Key Establishment
  - Data Integrity

Transport Layer Protocol - Data Integrity

- Data Integrity – SSH uses Message Authentication Code (MAC) algorithms to for data integrity check.

Transport Layer Protocol - Server Authentication

- The client initiates a SSH connection with the server.
- Server listens to default port 22 (this port can be changed) for SSH connections. A server uses a public key to prove its identity to a client.
- At this point, the server identity is verified. There are two cases:
  - If the client is accessing the server for first time, client is asked to authenticate server manually by verifying public key of server. Once the key is verified, the server is added in *known_hosts* file in ~/.ssh directory on client machine. The *known_hosts* file contains the information about all the verified servers by the client.
  - If the client is not accessing the server for the first time, the server's identity is matched with previously recorded information in known_hosts file for verification.

Transport Layer Protocol - Session Key Establishment

- Session Key Establishment – After the server is verified, both the parties negotiate a **session key** using a version of the **Diffie-Hellman** algorithm.
- The generated session key is a shared symmetric key.
- Session keys are generated before user authentication so that usernames and passwords can be sent encrypted.
- These keys are generally replaced at regular intervals (say, every hour) during the session and are destroyed immediately after use.

Three sub-protocols of SSH - User Authentication Protocol

- User Authentication Protocol – This part of SSH authenticates the user to the server. The server verifies that access is given to intended users only. Many authentication methods are currently used such as, typed passwords, Kerberos, public-key authentication, etc.
- Authentication is done using SSH key pair. One is public key that is used to encrypt data and can be freely shared. The other one is private key that is used to decrypt data and is never shared with anyone.
- After symmetric encryption has been established, the authentication of the client happens as follows:

SSH - Authentication of the Client

- The client begins by sending an ID for the key pair it would like to authenticate with to the server.
- The server checks the authorized_keys file of the account that the client is attempting to log into for the key ID.
- If a public key with matching ID is found in the file, the server generates a random number and uses the public key to encrypt the number and sends this encrypted message.
- If the client has the correct private key, it will decrypt the message to obtain the random number that was generated by the server.
- The client combines the obtained random number with the shared session key and calculates the MD5 hash of this value.
- The client then sends this MD5 hash back to the server as an answer to the encrypted number message.
- The server uses the same shared session key and the original number that it sent to the client to calculate the MD5 value on its own. It compares its own calculation to the one that the client sent back. If these two values match, it proves that the client was in possession of the private key and the client is authenticated.

SSH Services

- Secure Command-Shell (Remote Logon) – It allows the user to edit files, view the contents of directories, and access applications on connected device.

- Systems administrators can remotely start/view/stop services and processes, create user accounts, and change file/directories permissions and so on.

- Secure File Transfer – SSH File Transfer Protocol (SFTP) is designed as an extension for SSH-2 for secure file transfer. In essence, it is a separate protocol layered over the Secure Shell protocol to handle file transfers. SFTP encrypts both the username/password and the file data being transferred. It uses the same port as the Secure Shell server, i.e. system port no 22.

SSH Services

- Port Forwarding (Tunneling) – It allows data from unsecured TCP/IP based applications to be secured. After port forwarding has been set up, Secure Shell reroutes traffic from a program (usually a client) and sends it across the encrypted tunnel to the program on the other side (usually a server). Multiple applications can transmit data over a single multiplexed secure channel, eliminating the need to open many ports on a firewall or router.