CS 4910: Intro to Computer Security

Network Security IV:
Firewalls

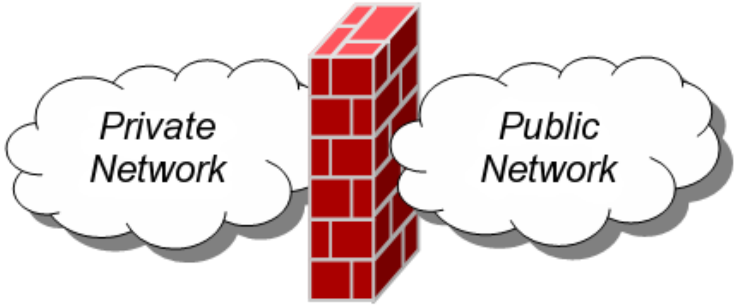
Instructor: Xi Tan

Today

- Network Security
  - Network Firewalls
  - Intrusion Detection Systems

Firewalls

- A **firewall** is an integrated collection of security measures designed to prevent unauthorized access to a networked computer system.
- A network firewall is similar to firewalls in building construction, because in both cases they are intended to isolate one "network" or "compartment" from another.



Firewalls

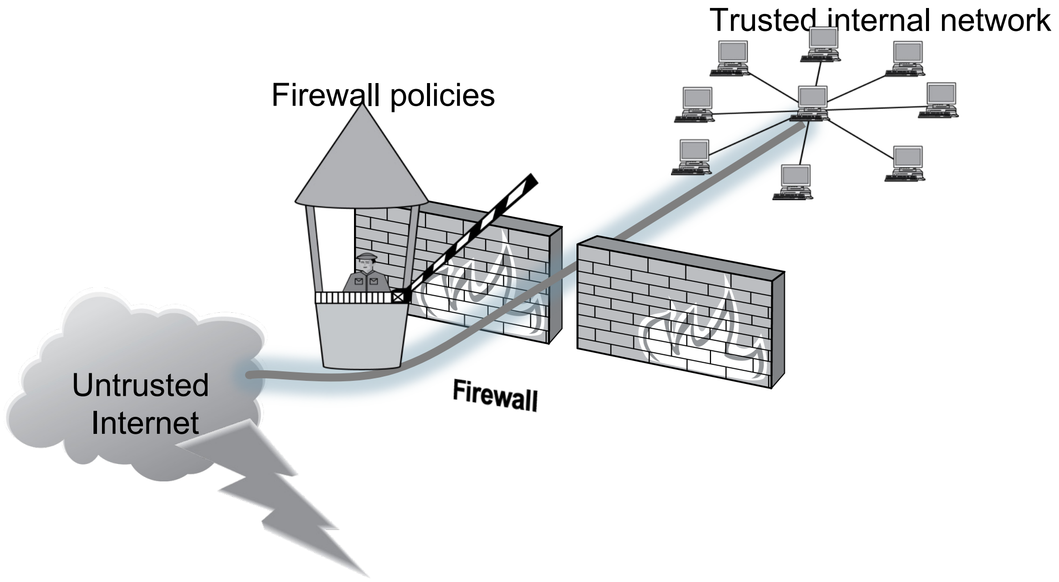
- What can we expect from a firewall?
  - single point that blocks unauthorized users from the protected network and simplifies security management
  - monitoring and reporting of security-related events
  - implementation of virtual private networks by means of IPsec, tunneling
  - convenient place for integration of other functions for network management
- A firewall does not protect against attacks that don’t go through the firewall
  - e.g., wireless connections, internal attacks, external devices connected directly to the internal machines/network

Firewalls

- Where can a firewall reside?
  - on a router
  - on a dedicated machine
  - personal firewall on a host
    - software that protects a single host rather than a network
    - e.g., Windows firewall, iptables in Linux, etc.
    - typically is configured to block most incoming traffic, but some applications can be let through
    - can be bypassed/disabled if host is compromised
- A firewall must be immune to penetration
  - ideally, it should run on a hardened system with a secured OS

Firewall Policies

- To protect private networks and individual machines from the dangers of the greater Internet, a firewall can be employed to filter incoming or outgoing traffic based on a predefined set of rules called **firewall policies**.



Policy Actions

- Packets flowing through a firewall can have one of three outcomes:
  - **Accepted:** permitted through the firewall
  - **Dropped:** not allowed through with no indication of failure
  - **Rejected:** not allowed through, accompanied by an attempt to inform the source that the packet was rejected
- Policies used by the firewall to handle packets are based on several properties of the packets being inspected, including the protocol used, such as:
  - TCP or UDP
  - the source and destination IP addresses
  - the source and destination ports
  - the application-level payload of the packet (e.g., whether it contains a virus).

Blacklists and WhiteLists

- **Blacklist** approach
  - All packets are allowed through except those that fit the rules defined specifically in a blacklist.
  - Flexible in ensuring that service to the internal network is not disrupted by the firewall
  - But naïve from a security perspective in that it assumes the network administrator can enumerate all of the properties of malicious traffic.
- **Whitelist** approach
  - A safer approach to defining a firewall ruleset is the default-deny policy, in which packets are dropped or rejected unless they are specifically allowed by the firewall.

Firewall Types

Packet filters (stateless)

If a packet matches the packet filter's set of rules, the packet filter will drop or accept it

"Stateful" firewalls

It maintains records of all connections passing through it and can determine if a packet is either the start of a new connection, a part of an existing connection, or is an invalid packet.

Application layer

It works like a **proxy** it can “understand” certain applications and protocols.

It may inspect the contents of the traffic, blocking what it views as inappropriate content (i.e. websites, viruses, vulnerabilities, ...)

Packet Filtering (stateless)

- simplest kind of firewall
- router has a list of access control rules
- router checks each received packet against security rules to decide whether to forward or drop it
- each rule specifies which packets it applies to based on a packet’s header fields
- can specify source and destination IP addresses, port numbers, protocol names, or wild cards
- actions are `ALLOW` or `DROP`
    - `<ACTION> <PTRCL> <SRC:PT> → <DEST:PT>`

Packet Filtering

- list of rules is examined one-by-one
- first matching rules determines how packet will be handled
- if no match is found, the default option can be to allow or drop
    - if the default option is drop, it is more noticeable to users
    - additional rules are added over time
    - this option, however, is preferred from security management point of view

Packet Filtering

- Policies based on IP header fields
  - a TCP or UDP service is specified by machine’s IP address and port number
    - e.g., web server `buffalo.edu` is at `128.205.201.56` port `80`
  - identify each service with triplet (`addr, prot, port`)
    - `addr` is machine’s IP address (a.b.c.d/[mask])
    - `prot` is TCP/UDP protocol identifier
    - `port` is the port number
  - example: all official web servers are located on subnet `12.34.56.x`
    - add (`12.34.56.0/24, TCP, 80`) to allowed list

Packet Filtering

- Let’s examine a sample ruleset

  -
    ```
    drop TCP *:* -> *:23
    allow * *:* -> *:*
    ```
  - what does it do?
    1. discard any TCP packet that is direct to port “23” (regardless of its source)
    2. allow any packet from any source to any destination
    3. firewall processes rules top-down
    4. thus, any TCP traffic directed to port 23 will be dropped, but all other traffic will be allowed
  - is this ruleset satisfactory?
    - there is no notion of a connection, inbound vs outbound connections
    - inbound and output packets to port 23 are dropped
    - default allow policy is undesirable

Packet Filtering

- Another example
  - assume that we want to allow
    - inbound connections to web server `12.34.56.78` on port `80`
    - all outbound connections
    - nothing else
  - we create the following ruleset

    -
      ```
      allow TCP *:* -> 12.34.56.78:80
      allow TCP (our-hosts):* -> *:*
      drop * *:* -> *:*
      ```
  - there are problems with it
    - TCP connections are bidirectional, data have to be able to go both ways

Packet Filtering

- Recall that TCP handshake is 3-way
  - send SYN, receive SYN-ACK, send ACK, then send data with ACK
- Suppose an inside host connects to an external machine on port 25 (mail)
  - initial packets get through (using rule 2)
  - SYN-ACK is dropped (fails the first two rules, matches the last)
- We need to distinguish between two types of inbound packets
  - allow inbound packets associated with an outbound connection
  - disallow inbound packets associated with an inbound connection

Packet Filtering

- We use TCP feature to make this distinction
  - ACK bit is set on all packets except the first one
  - recipients discard any TCP packet with ACK bit if it is not associated with an existing TCP connection
- Revised ruleset

  -
    ```
    allow TCP *:* -> 12.34.56.78:80
    allow TCP (our-hosts):* -> *:*
    allow TCP *:* -> (our-hosts):* (if ACK bit set)
    drop * *:* -> *:*
    ```
  - rules 1 and 2 permit inbound connections to 12.34.56.78 port 80
  - rules 2 and 3 allow outbound connections to any port

Packet Filtering

- Let’s see how our firewall stops packets
  - attacker wants to exploit finger service vulnerability (TCP port 79)
  - attempt 1: attacker sends SYN packet to internal machine
    - packet doesn’t have ACK bit set, so firewall rule drops it
  - attempt 2: attacker sends SYN-ACK packet to internal machine
    - firewall permits the packets, but then it is dropped by the TCP stack (i.e., ACK bit set, but it is not part of an existing connection)
- We can customize the ruleset to let any types of packets through according to the policy
- Does it mean we done now? how about spoofed addresses?

Packet Filtering

- Suppose an attacker can spoof source IP address and performs the following attack
  - let `12.34.56.77` be an internal host
  - attacker sends a spoofed TCP SYN packet from address `12.34.56.77` to another internal machine on port `79`
    - rule 2 in the ruleset allows the packet
  - target machine replies with SYN-ACK packet to `12.34.56.77` and waits for ACK (to finish handshake)
  - attacker sends spoofed ACK packet
  - attacker sends data packet(s)

Packet Filtering

- The attack above permits connections to internal hosts
  - it violates our security policy
  - it allows an attacker to exploit security vulnerabilities in internal machines
  - one difficulty: the attacker has to guess initial sequence number set by target in SYN-ACK packet to `12.34.56.77`
    - the attacker doesn’t see the response packet, but guessing might not be difficult
- What do we do now?
  - solve this by taking the interface a packet is coming from into consideration
  - mark a packet with interface id and incorporate ids into the rules

Packet Filtering

- New ruleset
  - internal interface is in, external interface out

  -
    ```
    allow TCP *:*/out -> 12.34.56.78:80/in
    allow TCP *:*/in -> *:*/out
    allow TCP *:*/out -> *:*/in (if ACK bit set)
    drop * *:* -> *:*
    ```
  - this allows inbound packets only to 12.34.56.78:80 (rule 1) or if ACK bit set (rule 3)
  - all other inbound packets are dropped
- Simple modification cleanly defeats IP spoofing threat
  - it simplifies ruleset administration (no need to hardcode internal hosts)

Stateless Firewalls

- Stateless packet filtering has its limitations
    - It does not remember the TCP connection states or any other information from any packet
    - It only looks at each packet
    - It cannot correlate packets.

Firewall Types

Packet filters (stateless)

Stateless packet filtering has its limitations

- It does not remember the TCP connection states or any other information from any packet
- It only looks at each packet
- It cannot correlate packets.

Stateful Firewalls

- Why need stateful: a stateless firewall doesn't know whether a packet belong to an acceptable connection

- **Stateful:**
  - Packet decision made in the context of a connection
  - If packet is a new connection, check against security policy
  - If packet is part of an existing connection, match it up in the state table & update table
    - Can be viewed as packet filtering with rules dynamically updated

Stateful Firewalls

- **Stateful firewalls** can tell when packets are part of legitimate sessions originating within a trusted network.
- Stateful firewalls maintain tables containing information on each active connection, including the IP addresses, ports, and sequence numbers of packets.
- Using these tables, stateful firewalls can allow only inbound TCP packets that are in response to a connection initiated from the internal network.

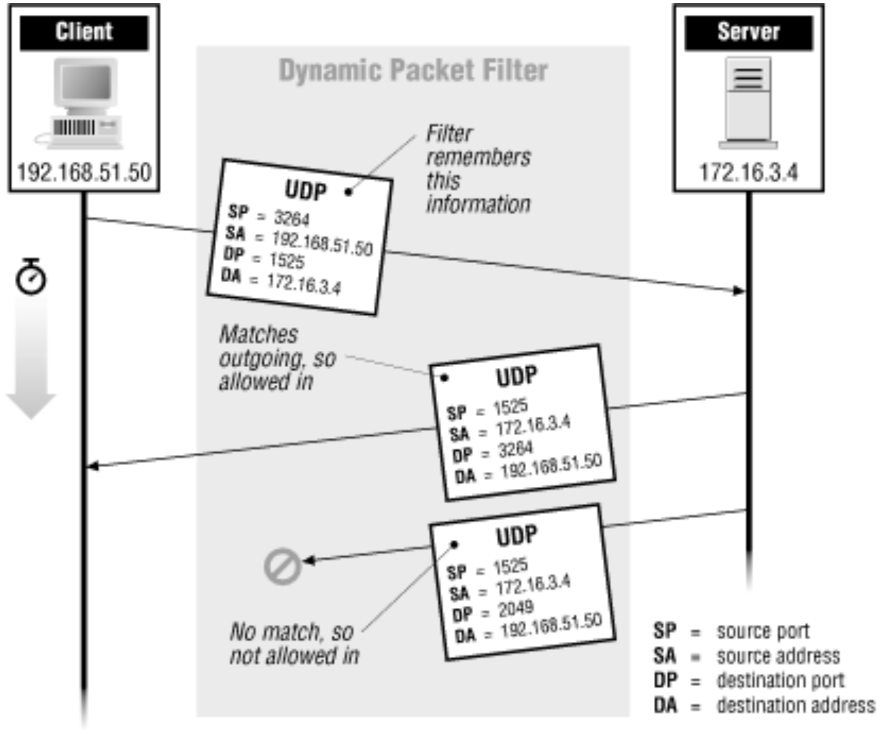
Stateful Firewalls

- Stateful packet inspection
  - packet decision is made in the context of a connection
  - if a packet is a new connection, check against security policy
  - if a packet is part of an existing connection, find it in the state table and update the table
    - this can be viewed as packet filtering with rules dynamically updated
- Example connection state stable

source address	source port	dest address	dest port	conn state
219.22.123.32	2112	124.33.44.5	80	established
124.33.44.129	1030	132.65.89.2	80	established
124.33.44.7	1035	190.3.15.4	25	established

Stateful Firewall Example

- Allow only requested UDP response:



Example from Packet Filtering docstore.mik.ua/orelly/networking_2ndEd/fire/ch08_01.htm

Firewall Types

Packet filters (stateless)

Stateless packet filtering has its limitations

- It does not remember the TCP connection states or any other information from any packet
- It only looks at each packet
- It cannot correlate packets.

"Stateful" firewalls

Advantages

- Can do everything a packet filter can do plus...
- Keep track of ongoing connections

Disadvantages

- Cannot see application data
- Slower than packet filtering

Application Layer Firewalls

- Application layer firewalls (or proxy firewalls)
    - is used as a relay for connections: Client ⟷ Proxy ⟷ Server
    - understands specific applications
        - limited versions of applications are available
        - proxy “impersonates” both sides of a connection
        - tends to be more secure than simple packet filters (can block application-specific attacks, can support authentication)
    - is resource-intensive (i.e., one process per connection)
    - certain proxies (e.g., HTTP) may cache data (e.g., web pages)

Firewall Types

Packet filters (stateless)

Stateless packet filtering has its limitations

- It does not remember the TCP connection states or any other information from any packet
- It only looks at each packet
- It cannot correlate packets.

"Stateful" firewalls

Advantages

- Can do everything a packet filter can do plus...
- Keep track of ongoing connections

Disadvantages

- Cannot see application data
- Slower than packet filtering

Application layer

Advantages

- Complete view of connections and applications data
- Filter bad data at application layer (viruses, Word macros)
  - Email system – disable attachment

Disadvantage

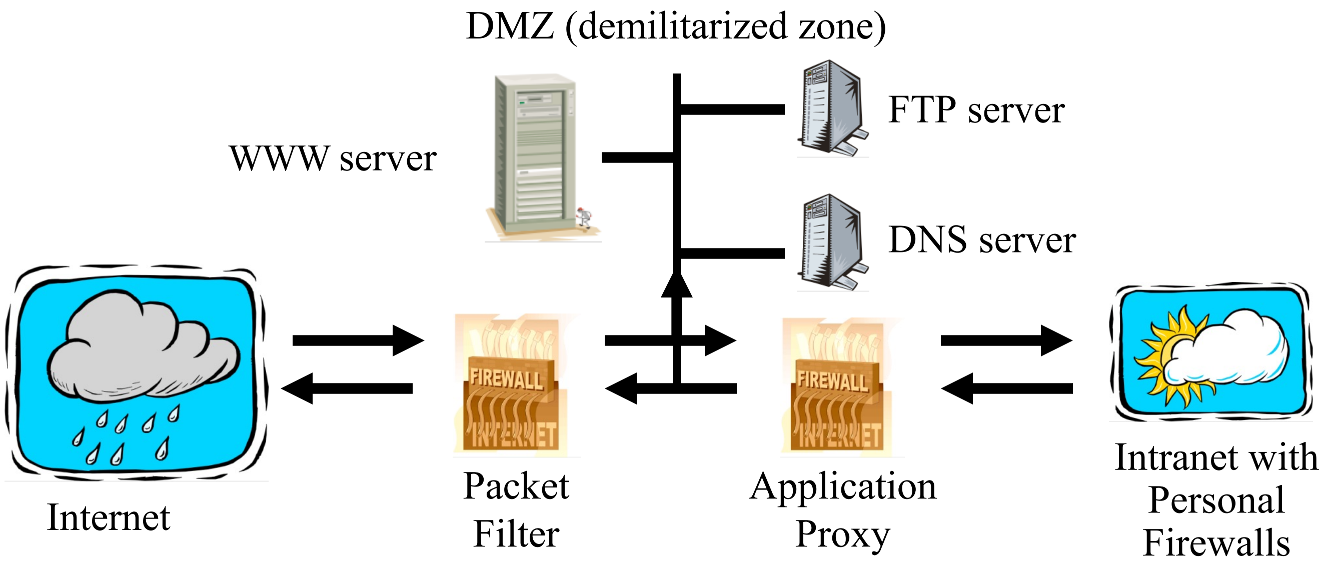
- Speed

Personal Firewalls

- Running on one PC, controlling network access
  - Windows firewall, iptables (Linux), ZoneAlarm, etc.
- Typically determines network access based on application programs
- Typically block most incoming traffic, harder to define policies for outgoing traffic
- Can be bypassed/disabled if host is compromised

Firewalls and Defense in Depth

- Example security architecture



Other tools

Tunnels: A technique that encapsulates one network protocol within another, creating a secure pathway for data.

- **Purpose:** To protect data as it travels across untrusted networks by isolating the payload from potential threats.
- **Key Features:** Encapsulation, secure pathway, protocol layering
- **Use Cases:** Forming the basis for VPN connections, Secure communication channels

SSH (Secure Shell): A protocol for secure remote access that encrypts data exchanged between the client and server.

- **Purpose:** To provide secure remote command-line access, file transfers, and even secure tunneling for other applications.
- **Key Features:** Encryption, authentication, port forwarding, file transfer (SCP/SFTP)
- **Use Cases:** Remote system administration, Secure file transfer, Creating encrypted tunnels

IPSec (Internet Protocol Security): A suite of protocols designed to secure IP communications by authenticating and encrypting each packet.

- **Purpose:** To protect data flows between devices over IP networks by ensuring confidentiality, integrity, and authentication.
- **Key Features:** Packet encryption, Authentication, Integrity verification, Network layer security
- **Use Cases:** Establishing secure VPN tunnels, Protecting data transmissions, Securing corporate networks

VPN (Virtual Private Network): A service that creates an encrypted connection (tunnel) over a public network, effectively extending a private network.

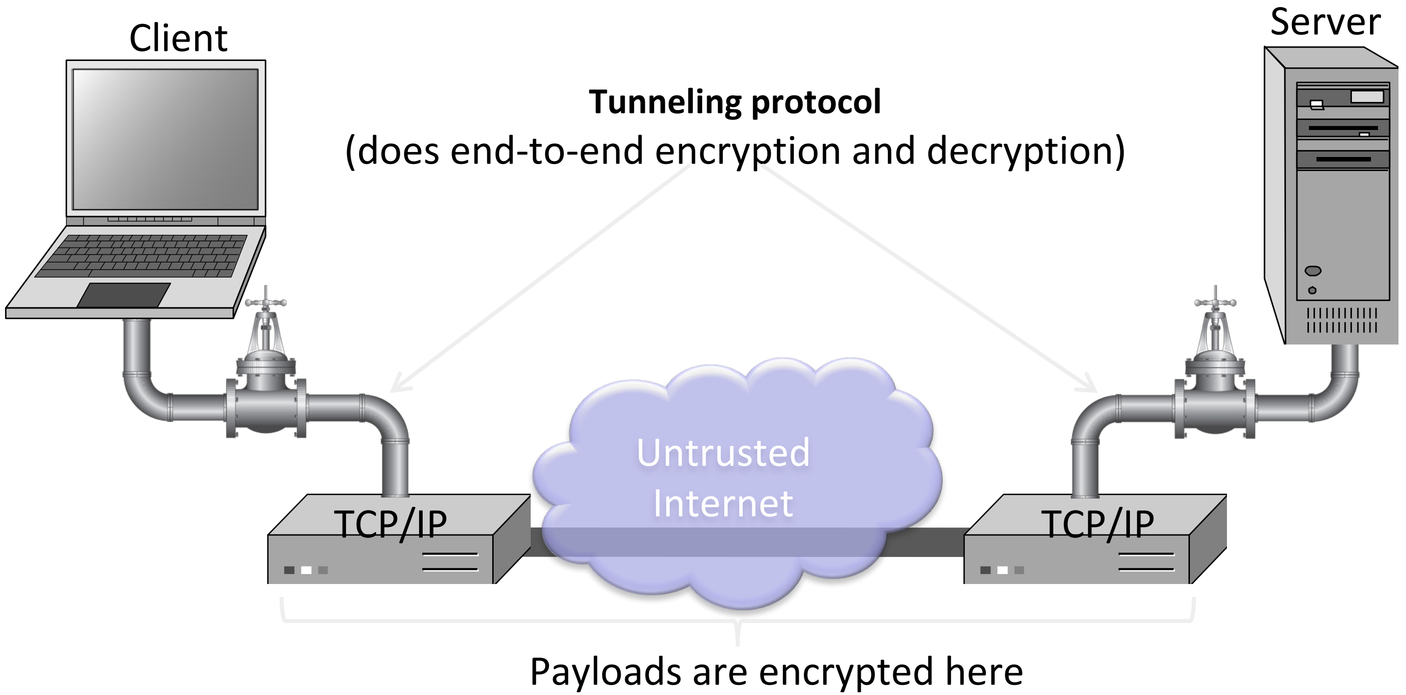
- **Purpose:** To allow remote users and branch offices to securely access a corporate network over the Internet.
- **Key Features:** Encrypted tunnel, Remote access, IP masking, Secure connectivity
- **Use Cases:** Secure remote work, Protecting data on public Wi-Fi, Connecting multiple office locations

Tunnels

- The contents of TCP packets are not normally encrypted, so if someone is eavesdropping on a TCP connection, he can often see the complete contents of the payloads in this session.
- One way to prevent such eavesdropping without changing the software performing the communication is to use a **tunneling protocol.**
- In such a protocol, the communication between a client and server is automatically encrypted, so that useful eavesdropping is infeasible.

Tunneling Prevents Eavesdropping

- Packets sent over the Internet are automatically encrypted.


Client
Server
Tunneling protocol
(does end-to-end encryption and decryption)
Untrusted Internet
TCP/IP
TCP/IP
Payloads are encrypted here

Secure Shell (SSH)

- **To establish an SSH connection (for example, telnet, FTP), a client and server go through the following steps:**

1. The client connects to the server via a TCP session.
2. The client and server exchange information on administrative details, such as supported encryption methods and their protocol version, each choosing a set of protocols that the other supports.
3. The client and server initiate a secret-key exchange to establish a shared secret session key, which is used to encrypt their communication (but not for authentication). This session key is used in conjunction with a chosen block cipher (typically AES, 3DES) to encrypt all further communications.

Secure Shell (SSH)

4. The server sends the client a list of acceptable forms of authentication, which the client will try in sequence. The most common mechanism is to use a password or the following public-key authentication method:

a) If public-key authentication is the selected mechanism, the client sends the server its public key.

b) The server then checks if this key is stored in its list of authorized keys. If so, the server encrypts a challenge using the client's public key and sends it to the client.

c) The client decrypts the challenge with its private key and responds to the server, proving its identity.

5. Once authentication has been successfully completed, the server lets the client access appropriate resources, such as a command prompt.

IPSec

- IPSec defines a set of protocols to provide confidentiality and authenticity for IP packets
  - Completely transparent to applications
- Each protocol can operate in one of two modes, **transport mode** or **tunnel mode.**
  - In **transport mode,** additional IPsec header information is inserted before the data of the original packet, and only the payload of the packet is encrypted or authenticated.
  - In **tunnel mode**, a new packet is constructed with IPsec header information, and the entire original packet, including its header, is encapsulated as the payload of the new packet.
    - Normally used for VPNs

Virtual Private Networking (VPN)

- **Virtual private networking (VPN)** is a technology that allows private networks to be safely extended over long physical distances by making use of a public network, such as the Internet, as a means of transport.
- VPN provides guarantees of data confidentiality, integrity, and authentication, despite the use of an untrusted network for transmission.
- There are two primary types of VPNs, **remote access VPN** and **site-to-site VPN.**

Types of VPNs

- **Remote access** VPNs allow authorized clients to access a private network that is referred to as an **intranet.**
  - For example, an organization may wish to allow employees access to the company network remotely but make it appear as though they are local to their system and even the Internet itself.
  - To accomplish this, the organization sets up a VPN endpoint, known as a **network access server, or NAS.** Clients typically install VPN client software on their machines, which handle negotiating a connection to the NAS and facilitating communication.
- **Site-to-site** VPN solutions are designed to provide a secure bridge between two or more physically distant networks.
  - Before VPN, organizations wishing to safely bridge their private networks purchased expensive leased lines to directly connect their intranets with cabling.