

# CSE 3231

# Computer Networks

## Network Security

William Allen, PhD

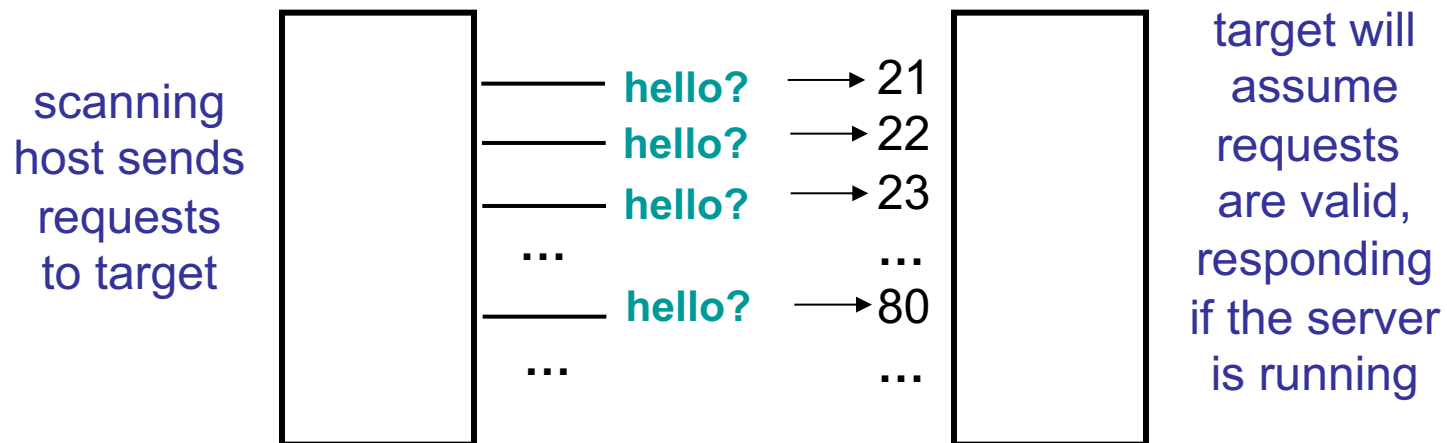
Spring 2022

# Network Reconnaissance

- What app is running and is it vulnerable?
  - port scans – discover active applications
  - are those applications vulnerable to attack?
- OS / application fingerprinting
  - look for identifying characteristics
  - find vulnerable OS or apps before attacking
- Find details and specifications in published documentation
  - bug reports, source code, RFCs

# Port Scans

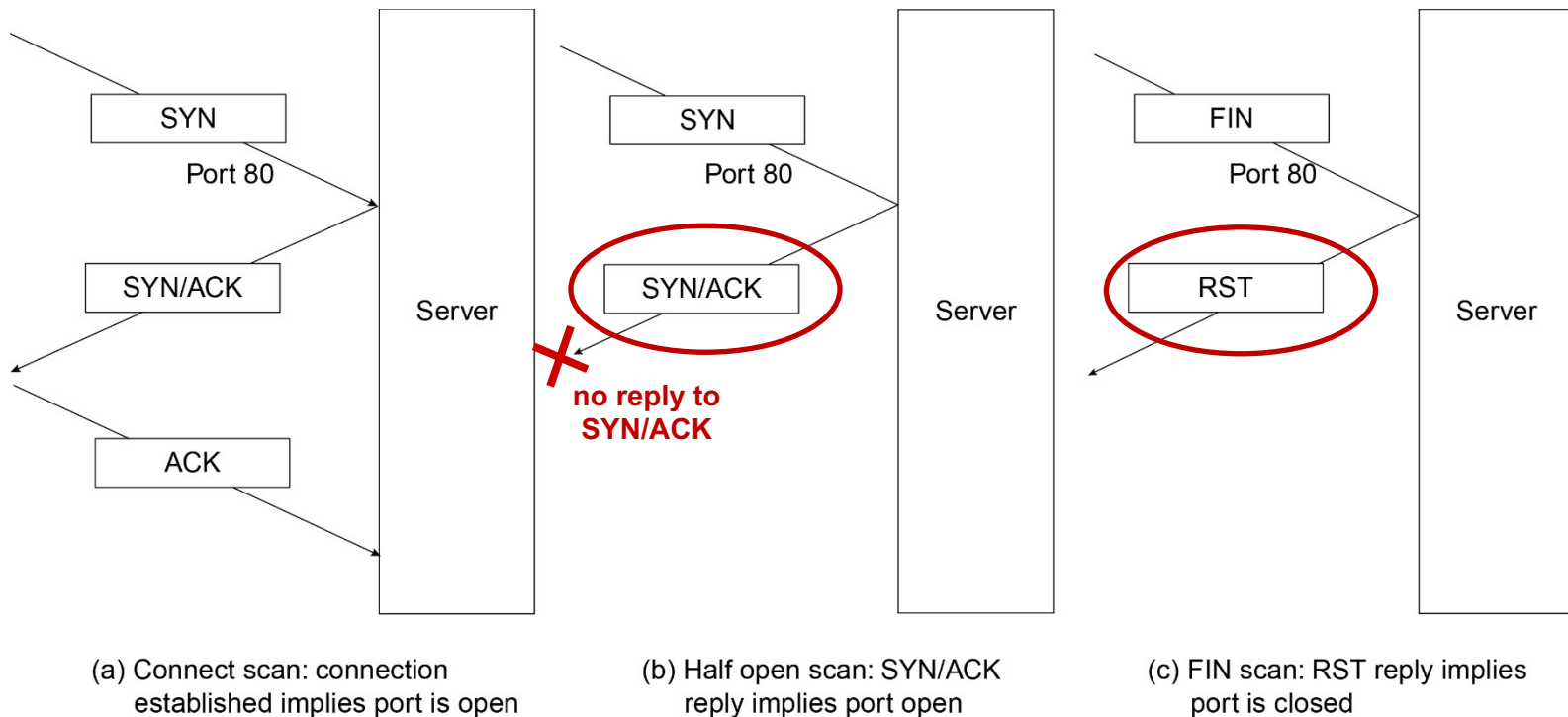
- To discover if a host is running a particular service (www, email, ssh, ftp, etc.) you can attempt to make a connection to each port and then see if it responds or ignores your request.
- Positive responses mean the server is running, thus an attacker can search for potential targets by seeing which hosts are running server applications.



# nmap

- **Nmap**: network security scanner
  - open source
  - transmits a variety of packets to detect ports, services, operating systems, etc.
  - test firewalls and ID systems
  - scales from one host to large networks
  - can scan in fast, slow, stealthy modes
  - some scans require admin, others do not

# Basic port scanning techniques



a) Connect scan

b) Half-open scan

c) FIN scan

# Nmap Port Scan Example

% **nmap scanme.nmap.org**

Starting Nmap 7.40 ( <https://nmap.org> ) at 2020-11-30 15:01 EST

Nmap scan report for scanme.nmap.org (45.33.32.156)

Host is up (0.088s latency).

Not shown: 992 closed ports

PORT	STATE	SERVICE
------	-------	---------

22/tcp	open	ssh
--------	------	-----

25/tcp	filtered	smtp
--------	----------	------

80/tcp	open	http
--------	------	------

135/tcp	filtered	msrpc
---------	----------	-------

139/tcp	filtered	netbios-ssn
---------	----------	-------------

445/tcp	filtered	microsoft-ds
---------	----------	--------------

9929/tcp	open	nping-echo
----------	------	------------

31337/tcp	open	Elite
-----------	------	-------

Nmap done: 1 IP address (1 host up) scanned in 9.34 seconds

# OS Fingerprinting

- What Operating System does a target server run on?
  - Windows?, Linux?, Mac OS?, Solaris?
    - maybe it has a new vulnerability that I can use to break into it, so how do I find out?
  - Protocol specifications are interpreted by OS designers/coders, each OS may respond in a slightly different way to requests
  - If we know those differences, we can use that information to figure out which OS is running!

# OS Fingerprinting Example

## Nmap scan report for scanme.nmap.org (74.207.244.221)

Initiating Ping Scan at 11:14

Scanning scanme.nmap.org (74.207.244.221) [1000 ports]

Discovered open port 22/tcp on 74.207.244.221

Discovered open port 80/tcp on 74.207.244.221

Initiating Service scan at 11:15

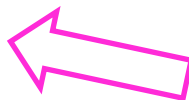
Not shown: 997 closed ports

22/tcp open ssh **OpenSSH 5.3p1** **Debian 3ubuntu7** (protocol 2.0)

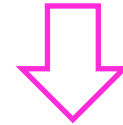
80/tcp open http **Apache httpd 2.2.14** ((**Ubuntu**))

1720/tcp filtered H.323/Q.931

Service Info: **OS: Linux**



Applications reported which  
versions they are running.





# Traceroute Example

*Traceroute* (Windows *tracert*) shows the series of routers between the source (*redacted*) and destination nodes

```
%traceroute qwest.net
```

```
traceroute to qwest.net (204.147.80.94), 30 hops max, 60 byte packets
```

```
1  homeportal (---.---.---.---) 3.632 ms 11.634 ms 48.640 ms
2  ---.---.---.---.lightspeed.sbcglobal.net (---.---.---.---) 48.757 ms 49.740 ms 49.867 ms
3  99.168.25.50 (99.168.25.50) 48.082 ms 49.084 ms 49.141 ms
4  12.123.34.242 (12.123.34.242) 55.429 ms 56.782 ms 67.020 ms
5  12.122.28.125 (12.122.28.125) 69.111 ms 69.027 ms 67.522 ms
6  attga402igs.ip.att.net (12.122.117.121) 71.848 ms 37.258 ms 40.143 ms
7  4.68.62.225 (4.68.62.225) 33.800 ms 35.214 ms 36.701 ms
8  ae-1-3501.edge6.Atlanta2.Level3.net (4.69.219.146) 37.724 ms 38.924 ms 40.505 ms
9  CenturyLink-level3-Atlanta2.Level3.net (4.68.62.82) 41.591 ms 35.792 ms 36.787 ms
10 min7-svcs-13.inet.qwest.net (205.171.177.86) 65.642 ms 69.006 ms 69.667 ms
11 www.qwest.net (204.147.80.94) 73.075 ms 74.310 ms 73.537 ms
```

# How Does Traceroute Work?

- *Traceroute* sends a series of *ping* (ICMP Type 8, Code 0) packets to the intended destination
  - However, the ping packets have small TTL values, starting at 1 and increasing by 1 for each subsequent ping packet in the series
- When the TTL value of each packet reaches 0, the router at that point sends an ICMP "Time Exceeded" message (ICMP Type 11, Code 0) back to the original sender, along with the router's IP address
- This allows Traceroute to get the sequence of routers along the path to the destination

# netcat (nc)

- networking utility
  - <http://netcat.sourceforge.net/>
- open source
- can make a TCP connection to any port
- can “act” as a client to test a server or provide simple data transmission (can send UDP too)
- built-in port-scanning capabilities
- get hexdump of transmitted or received data
- works on most Unix/Linux/MacOS platforms
  - Windows port is also available

# Packet Capture & Display

- There are several tools for capturing network packets for analysis & storage
  - **libpcap** - “**li**brary for **p**acket **cap**ture”, library of code that copies/filters network packets, supporting analysis and monitoring
  - **tcpdump** - linux/unix/Win tool that captures and stores packets directly from the network
    - tcpdump uses libpcap to capture packets
  - **Wireshark** - a graphical interface that makes it easier to view/filter captured traffic

# tcpdump

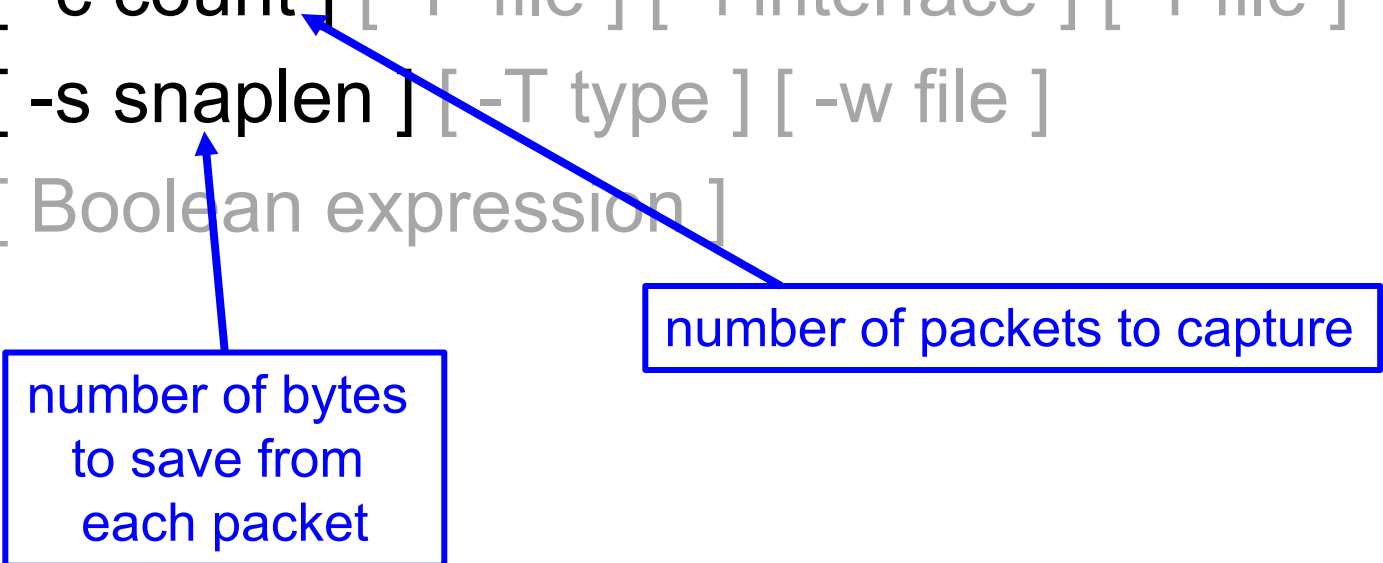
- UNIX/Linux tool that collects network data and displays it in specified format
  - Based on libpcap
- May be run “live” on a specified interface –  
*Note: this may be illegal (wiretapping)*
- Provides real-time filtering capabilities
- Can read data from a file that has previously been saved using TCPDump
  - Windows versions WinDump, WinPcap

# tcpdump

- *linux manual page for tcpdump:*
  - **tcpdump** - dump traffic on a network:
  - Options: **tcpdump** [ -adeFlnNOpqStvx ]  
[ -c count ] [ -F file ] [ -i interface ] [ -r file ]  
[ -s snaplen ] [ -T type ] [ -w file ]  
[ Boolean expression ]
  - DESCRIPTION:** Tcpdump prints out the headers of packets on a network interface that match the Boolean expression

# tcpdump

- *linux manual page for tcpdump:*
  - `tcpdump` - dump traffic on a network:
  - Options: `tcpdump [ -adeflnNOpqStvx ]`  
`[ -c count ] [ -F file ] [ -i interface ] [ -r file ]`  
`[ -s snaplen ] [ -T type ] [ -w file ]`  
`[ Boolean expression ]`



number of bytes  
to save from  
each packet

number of packets to capture

# tcpdump

- *linux manual page for tcpdump*:
  - **tcpdump** - dump traffic on a network:
  - Options: **tcpdump** [ -adeFlNOpqStvx ]  
[ -c count ] [ -F file ] [ -i interface ] [ -r file ]  
[ -s snaplen ] [ -T type ] [ -w file ]  
[ Boolean expression ]



file to save  
packets to



file to read  
packets from



# tcpdump

- *linux manual page for tcpdump:*
  - **tcpdump** - dump traffic on a network:
  - Options: **tcpdump** [ -adeFlnNOpqStvx ]  
[ -c count ] [ -F file ] [ -i interface ] [ -r file ]  
[ -s snaplen ] [ -T type ] [ -w file ]  
[ Boolean expression ]



name of file containing a  
predefined filter expression



network interface  
to capture from

# tcpdump

- *linux manual page for tcpdump:*
  - `tcpdump` - dump traffic on a network:
  - Options: `tcpdump [ -adeflnNOpqStvx ]`  
`[ -c count ] [ -F file ] [ -i interface ] [ -r file ]`  
`[ -s snaplen ] [ -T type ] [ -w file ]`  
`[ Boolean expression ]`



Boolean expression specifying  
the specific packets to capture

# Writing TCPdump Filters

- General format:
  - `<protocol header>[offset:length] <relation> <value>`
- Example: `tcpdump 'ip[9] = 1'` will select all IP packets that have protocol number 1 (ICMP) at byte 9 of the IP header
  - Single quote keeps UNIX shell from trying to interpret the filter
- You can also create a file, such as `/tmp/filter` and put “`ip[9] = 1`” in it.
  - Then execute: `tcpdump -F /tmp/filter`

# Bit Mask

- If you need to obtain values for fewer than 8 bits (byte), you can **&** with a bit mask
  - Example: `ip[0] & 0x0f` will ignore the first 4 bits of byte 0 (IP version/header length) and return just the IP header length
  - Thus '`ip[0] & 0x0f = 5`' will select all datagrams in which the header length is exactly 5 32-bit words or 20 bytes
  - or, '`ip[0] & 0x0f > 5`' will select all datagrams that have IP options

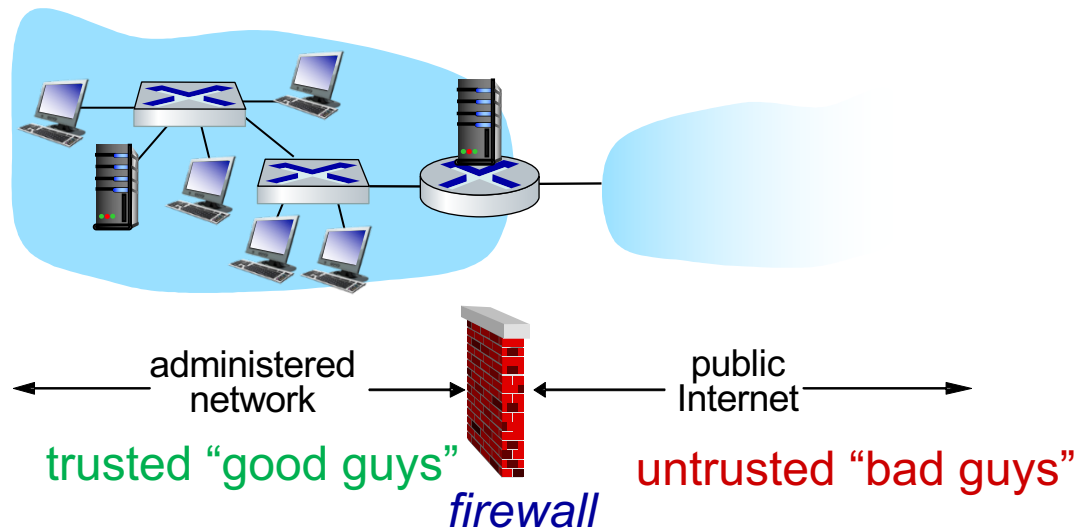
# Filter Example

- Catch all packets to broadcast address of `*.*.*.0` or `*.*.*.255`
  - (*byte 19 is the last byte of the IP address*)
  - `'ip[19] = 0x00 or ip[19] = 0xff'`
  - Alternatively `'ip[19] = 0 or ip[19] = 255'`
- To capture all *broadcast* packets *except* those from the 192.168.0.0 network
  - `'not src net 192.168 and (ip[19] = 0x00 or ip[19] = 0xff)'`

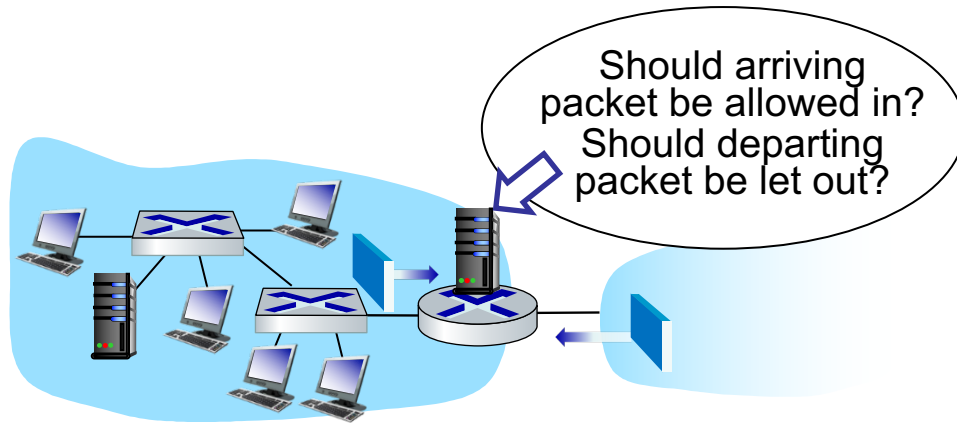
# Firewalls

## firewall

isolates organization's internal network from the Internet, allowing some packets to pass, blocking others



# Firewalls Filtering Traffic



- Firewall sits between internal network and the Internet, examines packets and makes a decision to forward/drop packets based on any of:
  - source or destination IP address
  - application (port #), protocol (TCP/UDP)
  - contents of one or more packets
  - known malicious traffic or behaviors

# Firewalls

- Firewall-based security depends on the firewall being the *only path* into the local network from outside
  - there should be no way to bypass the firewall via other gateways or wireless connections
- Firewalls may be used to create multiple *zones of trust*
  - A firewall can divide a network into a more-trusted zone internal to the firewall and a less-trusted zone external to the firewall

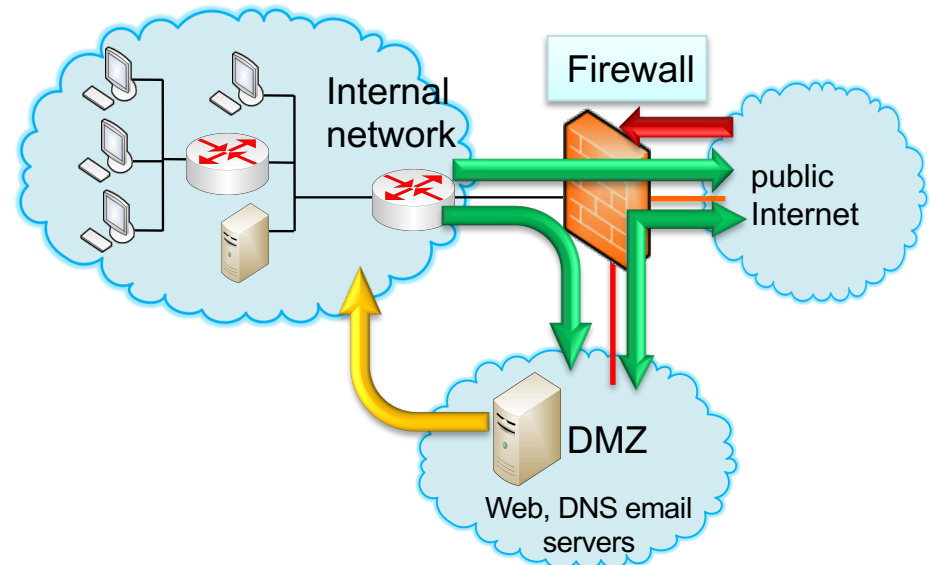


# Firewall Zones

- Common arrangements use 3 zones of trust:
  - the **internal** network,
  - an isolated area called the **DMZ** (DeMilitarized Zone)
  - and the rest of the **Internet**
- The DMZ holds services such as DNS and email servers that need to be publically accessible
  - The internal network and Internet can both access the DMZ, but hosts in the DMZ are limited in their access to the internal network to protect hosts in the network from compromised hosts in the DMZ

# Firewall Access Zones

- Allow/Block access based on IP address and/or port number
- **Allow:**
  - Internet to DMZ,
  - DMZ to Internet,
  - Internal to Internet
- **Block:**
  - Internet to Internal
- **Control:**
  - DMZ to Internal
- Common zones:
  - Internal Network (*Intranet*)
  - External Network (*Internet*)
  - Demilitarized Zone (*DMZ*)



# Firewall Deployment

Where can we exercise the access control?

## A. Network entry points

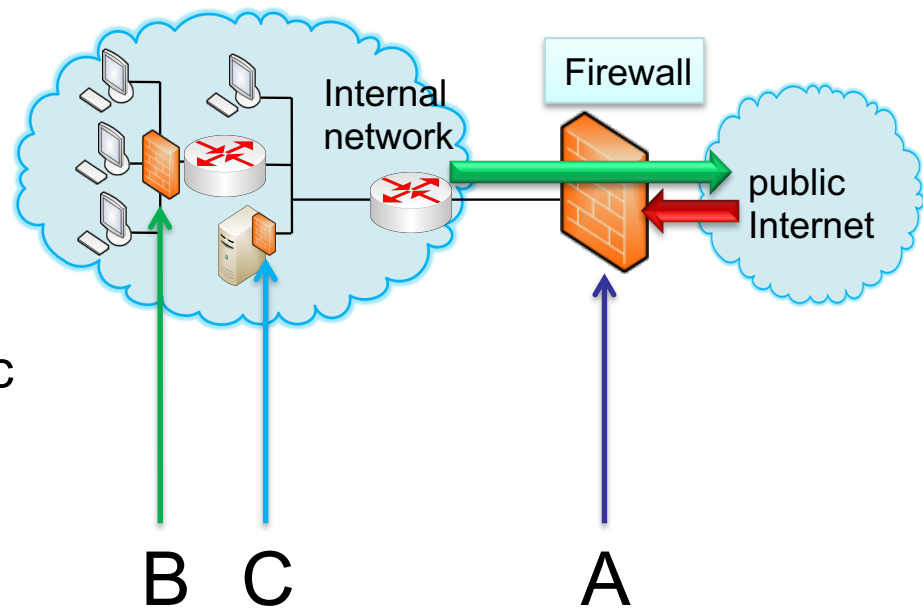
- Controlling external access

## B. Between subnets

- Controlling internal traffic

## C. Individual Hosts

- Controlling access to specific machines



# Purpose of Firewalls

prevent denial of service attacks:

- SYN flooding: attacker establishes many bogus TCP connections, no resources left for “real” connections

prevent illegal modification/access of internal data

- e.g., attacker compromises the company database and downloads/modifies private data

allow only authorized access to inside network

- pre-determined set of authenticated users/hosts

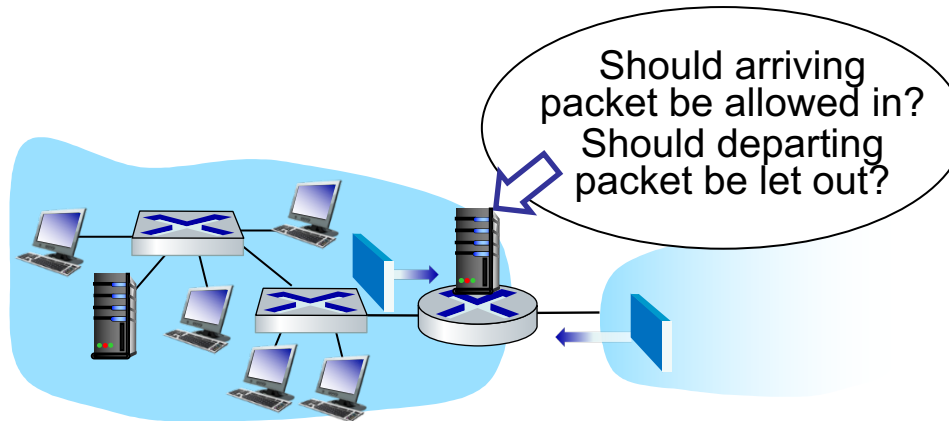
three types of firewalls:

- stateless packet filters
- stateful packet filters
- application gateways

# Stateless Packet Filtering

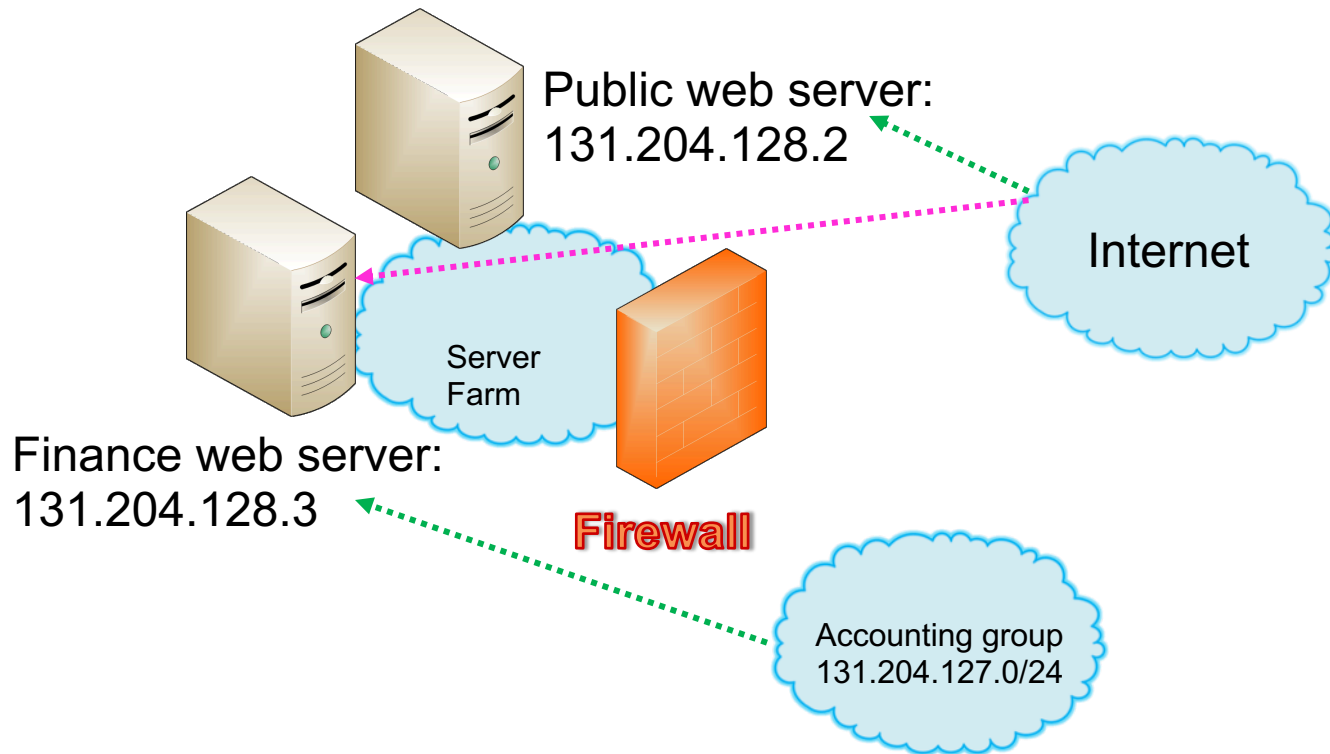
- Firewall decides whether *each* packet should proceed
  - Decision must be made on per-packet basis
  - No analysis of a packet's context in a network flow
    - does not keep track of a TCP connection or application
- Uses *packet header* for inspection
  - IP source and destination addresses, ports
  - Protocol identifier (TCP, UDP, ICMP, etc.)
  - TCP flags (SYN, ACK, RST, PSH, FIN)
  - ICMP message type
- Filtering rules are based on *binary pattern-matching*
- Cisco and other vendors may refer to their filtering rules as the firewall's *Access Control List* (ACL)

# Stateless Packet Filtering

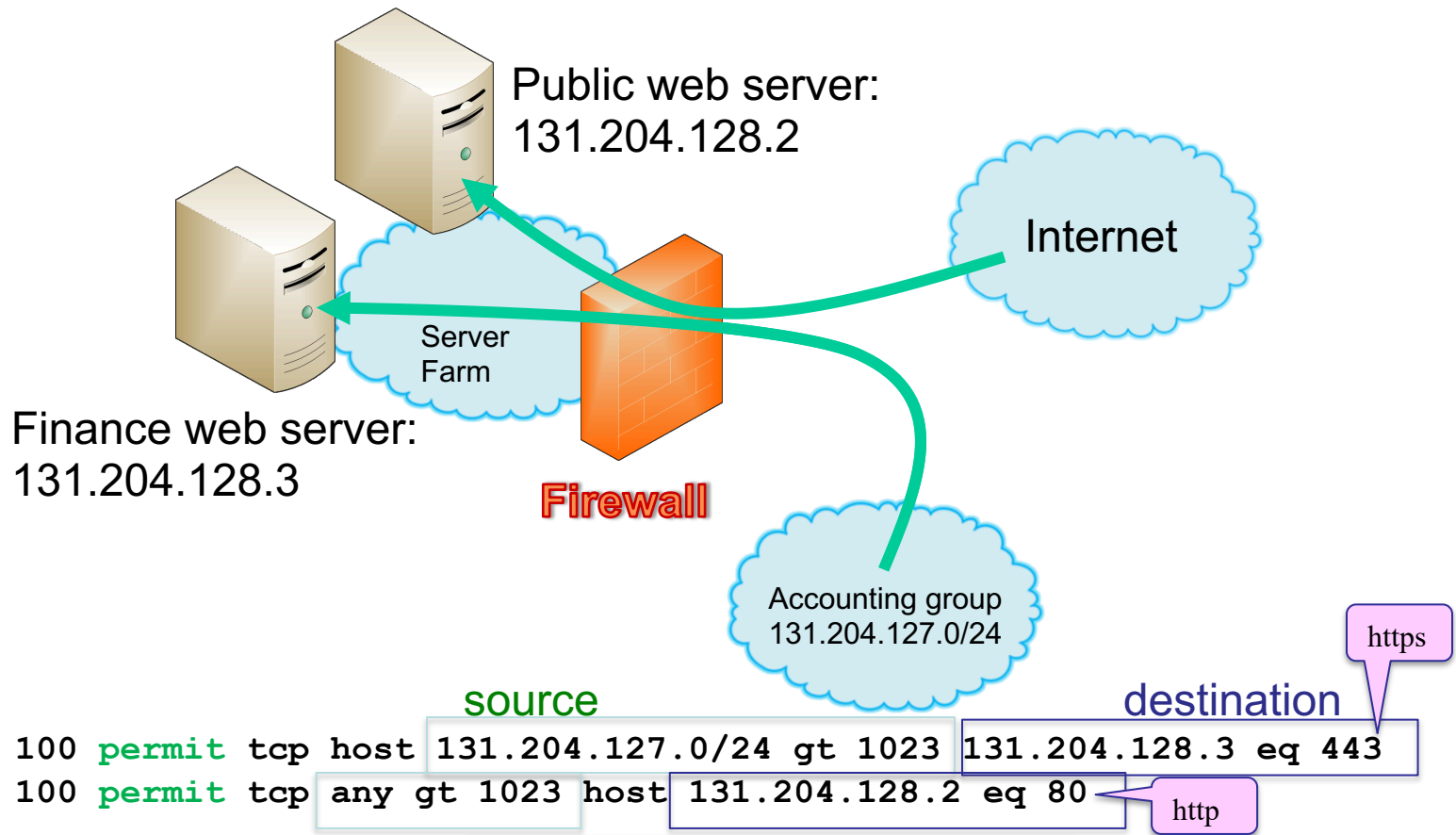


- **Example 1:** block *incoming* and *outgoing* datagrams with IP protocol field = 17 and any with either source or dest port = 23
  - **result:** all incoming, outgoing UDP flows and telnet (port 23) connections are blocked
- **Example 2:** block *inbound* TCP headers with SYN=1 & ACK=0
  - **result:** prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside servers

# Stateless Packet Filtering



# Stateless Packet Filtering



**access-list 100 deny ip any any**

Anything not explicitly permitted  
by the access list is **denied**!

Filtering is based on IP address and/or port number



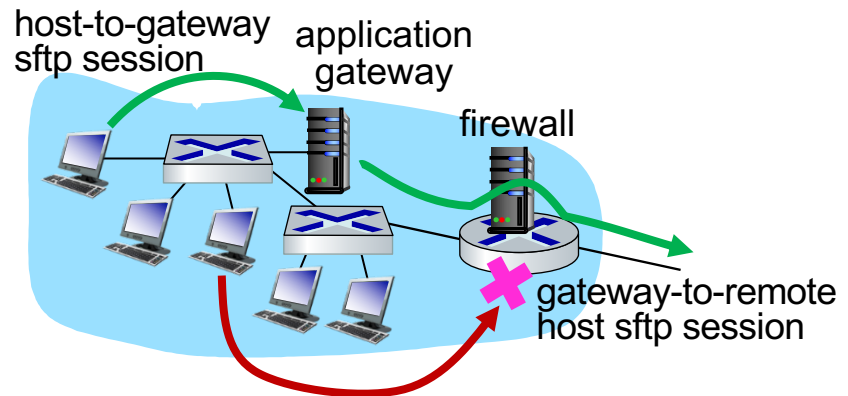
# Stateful Packet Filters

- Filtering is based on the *context* of a connection
  - If a new connection, then check against security policy
  - If existing connection, then look it up in the state transition table and update the table, if necessary
    - Example: only allow incoming traffic to a high-numbered port if there is an established connection to that port
- **Challenges**: stateless protocols
  - UDP and ICMP
- Default filter: **deny everything** not explicitly permitted
  - Block ICMP unless debugging is needed
- The filtering rules (ACL) have the same format as stateless filtering
  - Filters can be bypassed by using a VPN

# Application Gateways

- Filter packets on type of **application data** as well as on specific IP addresses or TCP or UDP ports

- *example:* allow select internal users to sftp outside



1. require all sftp users to connect through the gateway
2. for authorized users, the gateway sets up an sftp connection to the destination host and the gateway relays data between the two endpoints
3. the firewall's filter blocks all sftp connections not originating from the gateway (i.e., all unauthorized connections)

# Limitations of firewalls, gateways

- **IP spoofing:** firewall can't know if data “really” comes from claimed source
- if multiple apps need special treatment, each must have its own application gateway
  - filters often use all or nothing policy for UDP
  - client software must know how to contact gateway
    - e.g., must set IP address of proxy in Web browser
- *tradeoffs:* must balance the degree of communication with outside world with security requirements
  - many well-protected sites still suffer from attacks

# Limitations of Packet Filters

- Do not prevent application-specific attacks
  - No content (payload) inspection
  - For example, a firewall will not block an attack string that contains a buffer overflow in a URL
- No user authentication mechanisms
  - Only address-based authentication (spoofable)
  - Solution: list of addresses for each router interface
    - Packets with internal addresses should not come from outside the LAN (blocks the LAND attack)

# Outbound Filters

## Purpose of Outgoing Filters:

- Egress filtering prevents sending unauthorized traffic out to the Internet
  - Prevents leaking out traffic that is being used for internal configuration or network management
  - Stops leaks due to misconfiguration, as well as some network mapping attempts
  - Prevents internal systems from performing outbound IP spoofing attacks
  - Could prevent Trojans or Bots from contacting attackers to send private data out of the network

# Outgoing Firewall

- Network address translation devices can sometimes **leak out the private address space** located behind them. This can occur when
  - The device experiences high utilization or
  - Can be due to an attack
- If a private IP address leaks, an attacker may be able to use that information to **discover the layout** of your internal network
- Egress filtering can ensure that only packets sent from your **legal IP address range** are permitted out to the Internet

# Blocking ICMP

- **Block ICMP Echo-Replies** (type 0 code 0)
  - Echo-reply packets are returned by a system in response to receiving Echo-Request packets
  - This is usually an indication someone is using Ping
  - Echo-Reply packets can also be used as a covert communication channel.
    - For example, *Loki* uses Echo-Request/Echo-Reply packets to create a covert communication channel
- Inbound Echo-Requests can be blocked by default rules
  - You could also block outbound Echo-Replies to prevent internal hosts from answering

# ICMP Rules for a Network Firewall

- Block ICMP **Host Unreachable** (type 3 code 1)
  - Attackers can ID which systems are online simply by finding IP addresses that don't cause a **host unreachable** to be generated
  - The easiest way to solve this problem is to simply block the "host unreachable" reply packet from exiting the local network
- Block ICMP **Time Exceeded** (type 11 code 0)
  - Network mapping tools such as traceroute, tracert, Firewalk and tcptraceroute **map all of the routers** between a source and a target host by generating a series of packets with an artificially low Time To Live (TTL) value in the IP header
  - This causes the routers along the path to return ICMP **time exceeded** error messages, revealing paths in the network
  - Filtering outbound ICMP **time exceeded** messages blocks that



# Secure firewall settings

- Instead of blocking all ports, open only the ports that are necessary and block everything else
- Default:
  - Close all incoming ports (no servers/services for outside)
    - If a PC or a home small network do not need to provide services, such as http/https, then it is OK to close all incoming ports
  - Close all outgoing ports (destination ports)
- Then, open certain outbound (destination) ports
  - For example, ports 22 (SSH), 53 (DNS), 80 and 443 (WWW)
  - If SMTP or IMAP are used, port 25 or 143
  - Allow the response packets for outgoing packets to pass through the firewall

# Access Control Lists (ACL)

- An ACL consists of one or more entries
  - The ACL is a sequential collection of *permit* and *deny* conditions
  - Depending on the ACL type, one can specify a match criteria using the source and destination addresses, the protocol, the ports (for TCP or UDP), the ICMP type, ICMP code, or the Ethernet Type field

# Access Control Lists (ACL)

- The order of the entries is important
  - When a firewall decides whether to accept or refuse a connection, the firewall tests the packet against each ACL rule **in the order in which the entries are listed**
  - After the firewall finds the first match, it does not check any more entries
    - For example, if one creates an entry at the beginning of an ACL that explicitly permits all traffic, the firewall does not check any further statements in the ACL
  - Therefore, ACLs should have an implicit **deny** at the end, if there was no rule to allow the packet, it will be denied access to the network

# Access Control Lists (ACL)

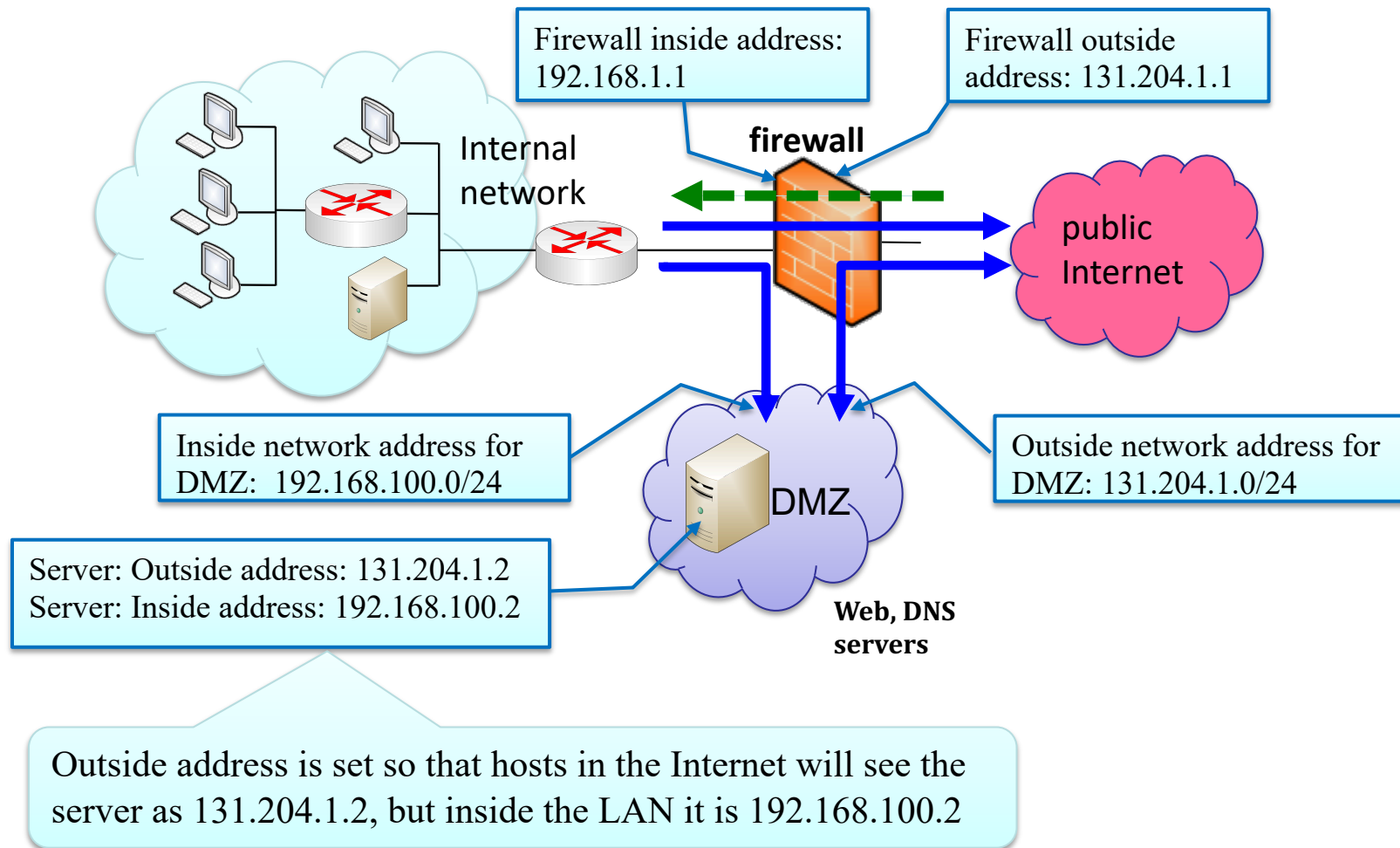
## Implicit Deny

- Some Firewalls implicitly “deny all” at the end of the list of rules, otherwise it must be added manually

```
access-list 100 permit tcp host 131.204.127.0/24 gt 1023 131.204.128.3 eq 443
access-list 100 permit tcp any gt 1023 host 131.204.128.2 eq 80
access-list 100 deny ip any any

interface Ethernet 0/0
access-list 100 in
```

# Example: Firewall Configuration



# Example: Cisco Firewall Rules

The screenshot shows the Cisco PIX Device Manager 3.0 interface. The 'Access Rules' tab is selected. The rules table is as follows:

#	Action	Source Host/Network	Destination Host/Network	Interface	Service	Log Level Interval	
-	✓	any	any	inside (outbound)	ip		Implicit o
-	✓	any	any	dmz (outbound)	ip		Implicit o
1	✓	any	192.168.100.0/24	outside	http/tcp		
2	✓	any	192.168.100.0/24	outside	domain/udp		

At the bottom, there are checkboxes for 'Allow traffic' (checked) and 'Deny traffic' (unchecked), along with 'Apply', 'Reset', and 'Advanced...' buttons. A status bar at the bottom indicates 'Running configuration successfully saved to flash memory.' and shows the user as '<admin>' with 15 connections, dated 11:17:03 UTC Mon Jul 12 2004.

All incoming packets are denied implicitly

Allow inside network to go anywhere

Allow DMZ network to go outside; Cisco does not allow DMZ to go inside implicitly

Allow outside network to go to DMZ using only http and DNS packets  
Everything else is denied for outside network

# Example: Cisco Firewall Access Control List (ACL)

- An ACL consists of one or more entries
  - The ACL is a sequential collection of permit and deny conditions
  - Depending on the ACL type, one can specify match criteria using the source and destination addresses, the protocol, the ports (for TCP or UDP), the ICMP type, ICMP code, or the EtherType
- The order of the entries is important
  - The firewall tests a packet against each ACL in the order in which the entries are listed to determine whether to pass or deny the packet
  - After it finds a match, the firewall does not check any more entries
    - If an entry at the beginning of an ACL explicitly permits all traffic, the firewall does not check any further statements in the ACL
  - All ACLs have an implicit deny entry at the end of the ACL
  - Unless one explicitly permits it, traffic cannot pass through the firewall
    - To allow all users to access a network through the firewall except for those with particular IP address, you would have to deny that IP address and allow all other IP addresses by default

# Example: Cisco Firewall Access Control List (ACL)

- The firewall tests packets against the conditions listed in the ACL one at a time
- The first match that is found determines whether the firewall accepts or rejects the packet
- Because some firewalls, like those from Cisco, will stop testing conditions after the first match, the order of the conditions is critical
  - Accepts are listed first and if none of those conditions match, the router will reject the packet because of an implicit “deny all” clause at the end



# Example: Cisco Firewall Firewall Rules

Source	Destination	Service	Action	
131.204.0.0/16	any	http	Permit	Allowed Actions
131.204.0.0/16	any	DNS/UDP	Permit	
131.204.0.0/16	any	https	Permit	
131.204.0.0/16	any	SMTP	Permit	
Source	Destination	Service	Action	
131.204.0.0/16	any	tcp-outbound	Deny	Denied Actions
131.204.0.0/16	any	udp/outbound-ports	Deny	

After the firewall processes the above ACL rules,  
there is an implicit deny for all other actions

# Summary - Firewalls

- Firewalls *cannot verify* that connections are authorized or determine the identity of a source
  - On the other hand, they don't require any cooperation or information from the source of a packet to decide whether to accept it
- They give the administrators of a network a certain level of control over what enters and what leaves their network

# Summary - Firewalls

- Firewalls also *cannot determine* if an authorized user has been compromised and will continue to pass that user's traffic
  - other network security components must determine if the user's traffic is malicious
- They also *cannot determine* if a machine inside the network is being used to pass information into or out of the network for malicious or illegal purposes
  - outgoing connections are usually allowed