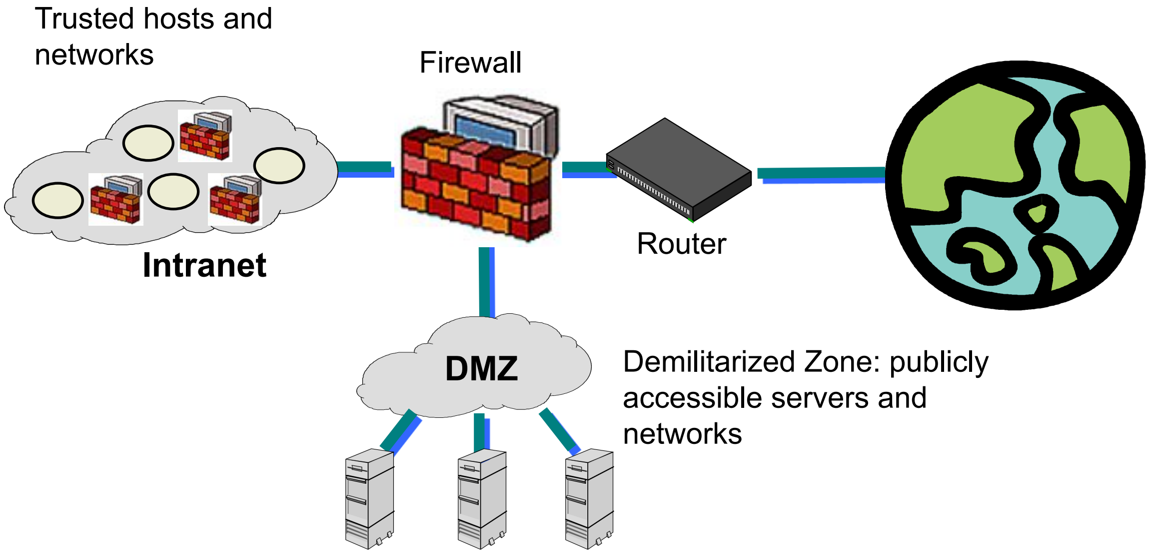
Firewalls

Course of Cybersecurity

Firewalls

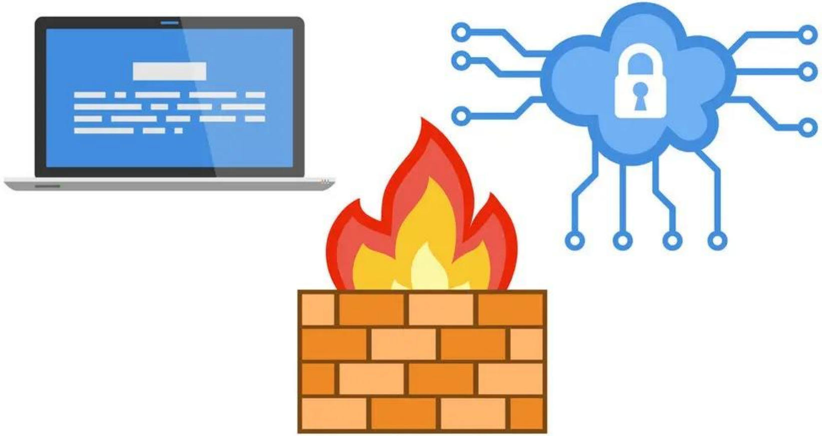
- Idea: separate local network from the Internet



Firewall

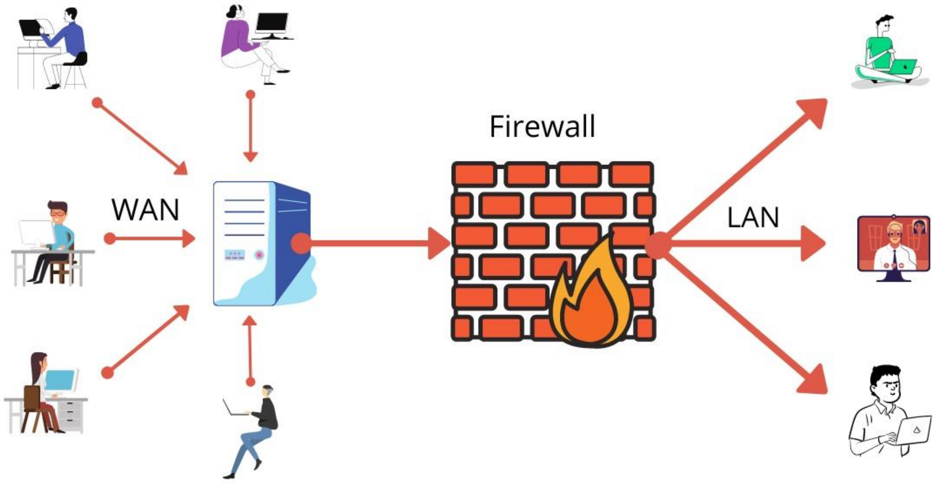
Firewall controls and monitors network traffic

- Most cases: a firewall links an internal network to the external world (public internet)
  - Limits the inbound and outbound traffic
  - Only authorized traffic passes the firewall
  - Hides the internal network to the external world
  - Controls and monitors accesses to service
- On end-user machines
  - “Personal firewall”
- Should be immune to attacks



Firewall

- Does not protect with respect to attacks that pass the firewall
- Does not protect from attacks originated within the network to be protected
    - Unless a personal firewall is used
- Is not able to avoid/block all possible viruses and worms (too many, dependent on specific characteristics of the Operating Systems)

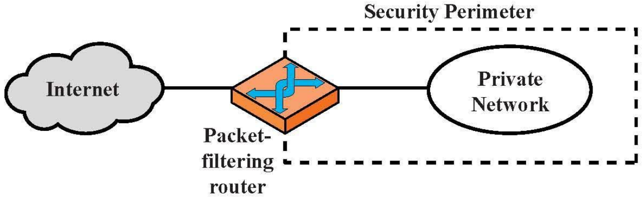


Firewall Types

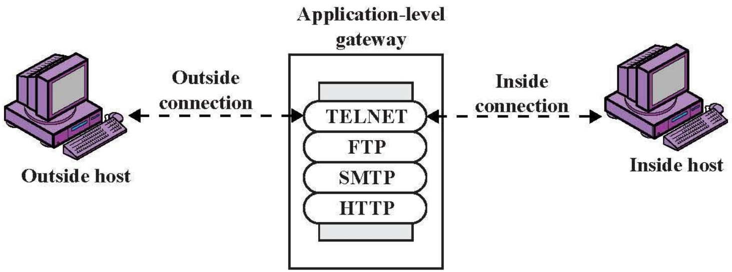
Packet- or session-filtering router (Packet filter)	Proxy gateway	Personal firewall with application-specific rules

• filtering is done by inspecting headers (and payloads, in some cases)
• usually stateless, sometimes stateful
	• All incoming traffic is directed to firewall, all outgoing traffic appears to come from firewall
	• Application-level: separate proxy for each application
	  • Different proxies for SMTP (email), HTTP, FTP, etc.
	  • Filtering rules are application-specific
	• Circuit-level: application-independent, “transparent”
		• E.g., no outbound telnet connections from email client

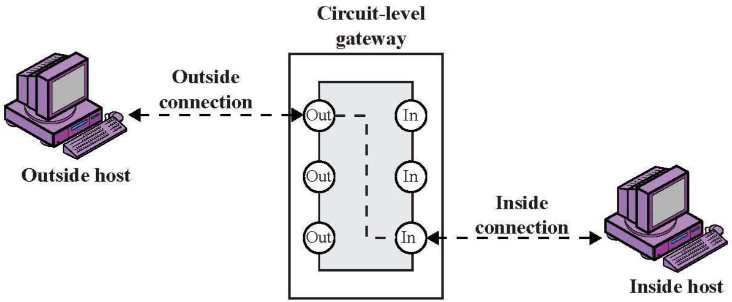
Firewall Types



(a) Packet-filtering router



(b) Application-level gateway



(c) Circuit-level gateway

Packet Filtering

For each packet, firewall decides whether to allow it to proceed

Decision must be made on per-packet basis

- Stateless; cannot examine packet's context (TCP connection, application to which it belongs, etc.)

To decide, use information available in the packet

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 pattern-matching

Default rule: accept/reject

Packet Filtering Examples

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

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

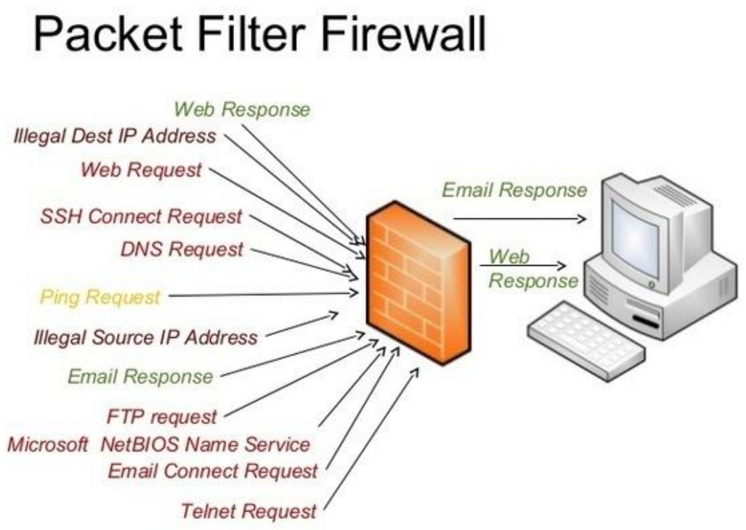
```
access-list 100 permit tcp any gt 1023 host 172.168.10.12 eq 21
access-list 100 permit tcp any gt 1023 host 172.168.10.12 eq 20
 ! Allows packets from any client to the FTP control and data ports
access-list 101 permit tcp host 172.168.10.12 eq 21 any gt 1023
access-list 101 permit tcp host 172.168.10.12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

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
!
```

Anything not explicitly permitted by the access list is denied!

Weaknesses of Packet Filters

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



Fragmentation Attacks

A fragmentation attack uses two or more pcks such that each pck passes the firewall; BUT when the pcks are assembled (and it is possible to check TCP header) they form a pck that should be dropped. Examples

Two ack pack assembled form a SYN pck (TCP request)

Split ICMP message into two fragments, the assembled message is too large

- Buffer overflow, OS crash

Fragment a URL or FTP "put" command

- Firewall needs to understand application-specific commands to catch

IP fragments overlap

- some operating systems do not properly handle that

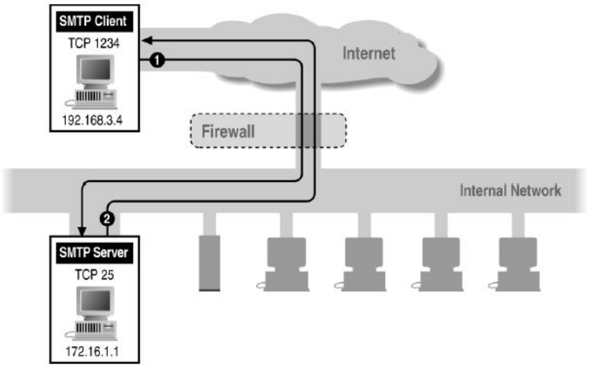
excessive number of incomplete fragmented datagrams

- denial of service attack or an attempt to bypass security measures

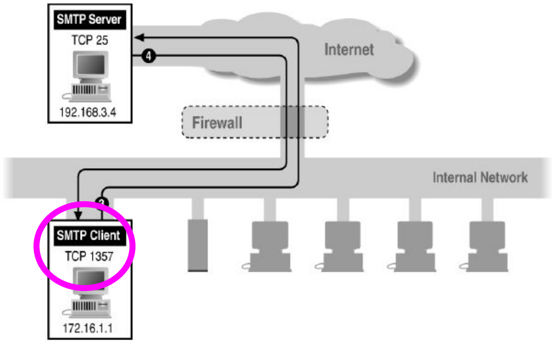
Limitation of Stateless Filtering

- In TCP/IP, well-known ports (<1024) are typically assigned to servers. Examples: 20–21 FTP, 23 Telnet, 25 SMTP, 80 HTTP
- Clients use ephemeral ports (>1023)
  - These ports are dynamically assigned
  - They must remain reachable to receive server responses
- Key problem for stateless firewalls: What should the firewall do when it receives an incoming packet destined to a client's ephemeral port, e.g. port 1234?
  - It might be a legitimate reply to a previously established connection
  - Or it might be a malicious inbound connection attempt
  - Without connection state, the firewall cannot distinguish between the two
- Fundamental limitation: A stateless firewall cannot reliably differentiate between return traffic and unsolicited incoming traffic, because it does not track connection state

Example: Variable Port Use



Inbound SMTP



Outbound SMTP

Session Filtering

- Decision is 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

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

iptables

- Iptables is used to set up, maintain, and inspect the tables of IPv4 packet filter rules in the Linux kernel.
- Several different tables may be defined. Each table contains several built-in chains and may also contain user-defined chains.
- chain = list of rules which can match a set of packets
  - each rule specifies criteria for a packet and an associated target, namely what to do with a packet that matches the pattern

tables

normally there exist a few standard tables

- filter (default)
- nat
- mangle
- raw

each table contains built-in chains and may contain user-defined chains

Built-in chains

- PREROUTING: Packets will enter this chain before a routing decision is made.
- INPUT: Packet is going to be locally delivered.
- FORWARD: All packets that have been routed and were not for local delivery will traverse this chain.
- OUTPUT: Packets sent from the machine itself will be visiting this chain.
- POSTROUTING: Routing decision has been made. Packets enter this chain just before handing them off to the hardware.
- Built-in chains have a *policy*, for example DROP, which is applied to the packet if it reaches the end of the chain.

targets

- each rule specifies criteria for a packet and a target
- if the packet does not match a rule, next rule in chain is then examined
- if it matches, then the next rule is specified by the value of the target
- targets: accept, drop, queue, return, or name of a user-defined chain

standard targets

- accept = let the packet through
- drop = drop the packet on the floor
- queue = pass the packet to userspace (what happens depends on a queue handler, included in all modern linux kernels)
- return = stop traversing this chain and resume at the next rule in the previous (calling) chain

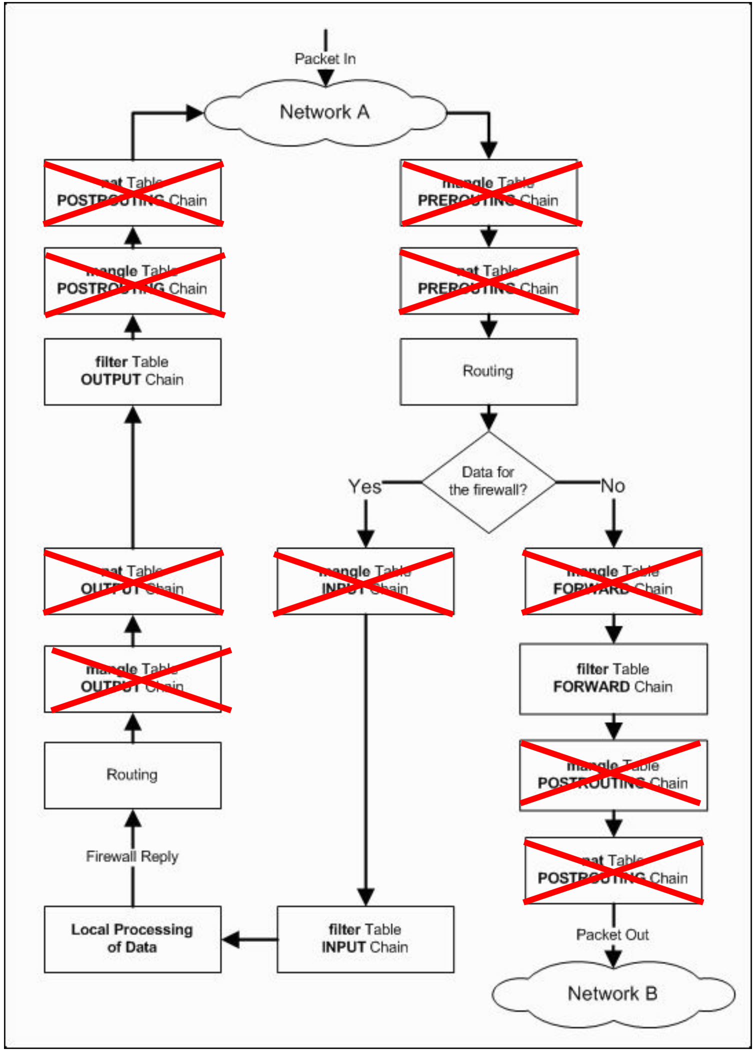
tables, again

- filter
  - default table, contains the built-in chains INPUT (for packets destined to local sockets), FORWARD (for packets being routed through the box), and OUTPUT (for locally-generated packets)
- nat
  - Network Address Translation occurs before routing. Facilitates the transformation of the destination IP address to be compatible with the firewall's routing table. Used with NAT of the destination IP address,
  - It consists of three built-ins: PREROUTING (for altering packets as soon as they come in), OUTPUT (for altering locally-generated packets before routing), and POSTROUTING (for altering packets as they are about to go out)

tables, again 2

- mangle
  - TCP header modification (modification of the TCP packet quality of service bits before routing occurs)
  - built-in chains: PREROUTING (for altering incoming packets before routing), OUTPUT (for altering locally-generated packets before routing), INPUT (for packets coming into the box itself), FORWARD (for altering packets being routed through the box), and POSTROUTING (for altering packets as they are about to go out)
- raw
  - mainly used for configuring exemptions from connection tracking

Iptables Packet Flow Diagram



iptables extended modules

- iptables can use extended packet matching modules
  - two ways: implicitly (when -p is specified) or explicitly (with the -m option, followed by the matching module name)
  - after these, various extra command line options become available, depending on the specific module.
- -m state [!] --state st
  - where st in {INVALID, ESTABLISHED, NEW, RELATED}
    - a new connection has state **related** if it's a new connection and if is related to an already established connection

extended modules

module name	info
account	accounts traffic for all hosts in defined network/netmask
addrtype	matches packets based on their address type (UNSPEC, UNICAST, LOCAL, BROADCAST, ANYCAST, MULTICAST…)
connbytes	matches by how many bytes or packets a connection have transferred so far, or by average bytes per packet
connlimit	allows to restrict the number of parallel TCP connections to a server per client IP address (or address block)
connrate	module matches the current transfer rate in a connection

extended modules

module name	info
conntrack	allows access to connection tracking information (more than the "state" match)
hashlimit	gives you the ability to express '1000 packets per second for every host in 192.168.0.0/16' or '100 packets per second for every service of 192.168.1.1' with a single iptables rule
icmp	allows specification of the ICMP type
iprange	matches on a given arbitrary range of IPv4 addresses
length	matches the length of a packet against a specific value or range of values

extended modules

module name	info
mac	matches source MAC address. It must be of the form XX:XX:XX:XX:XX:XX. Note that this only makes sense for packets coming from an Ethernet device and entering the PREROUTING, FORWARD or INPUT chains
multiport	matches a set of source or destination ports. Up to 15 ports can be specified. A port range (port:port) counts as two ports. It can only be used in conjunction with -p tcp or -p udp
nth	matches every 'n'th packet
owner	matches various characteristics of the packet creator, for locally-generated packets. It is only valid in the OUTPUT chain, and even this some packets (such as ICMP ping responses) may have no owner, and hence never match

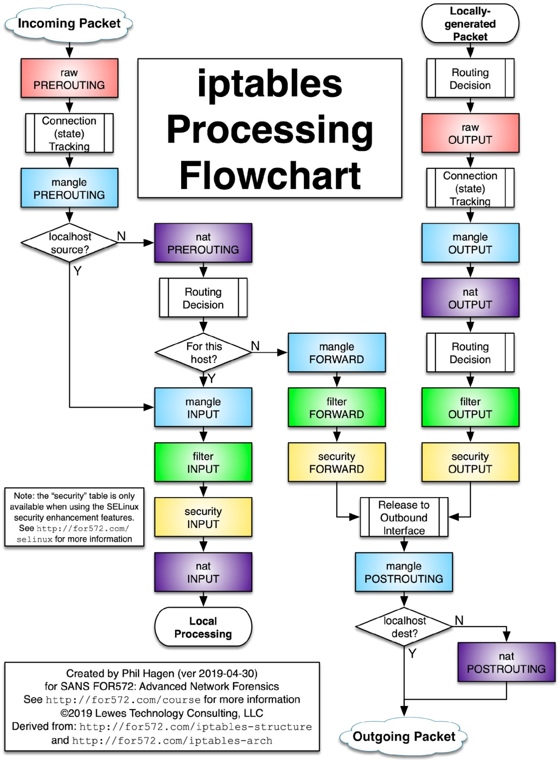
extended modules

module name	info
psd	attempts to detect TCP and UDP port scans. This match was derived from Solar Designer's scanlogd
quota	Implements network quotas by decrementing a byte counter with each packet
random	randomly matches a certain percentage of all packets
state	allows access to the connection tracking state for this packet
tcp	extensions are loaded if '--protocol tcp' is specified
time	matches if the packet arrival time/date is within a given range
ttl	matches the time to live field in the IP header
udp	loaded if '--protocol udp' is specified

many other modules!

iptables options

2019



type of options

- COMMANDS
  - -A (--append) chain rule-specification
  - -L (--list) [chain]
- PARAMETERS
  - [!] -p (--protocol) protocol
  - [!] -s (--source) address[/mask]
  - [!] -i (--in-interface) name
- OTHER
  - -n (--numeric)
  - -j (--jump) target

see man iptables for more

Firewall: session filter

Rules:

IPTABLES -t TABLE -A CHAIN -[I|O] IFACE -s x.y.z.w -d a.b.c.d -p PROT -m state --state STATE -j ACTION

Rules use

PACKET ADDRESS (TABLE) = nat | filter | …

ORIGIN OF CONNECTION/PACK. = INPUT (I) | OUTPUT (O)| FORWARD (F) | …

NETWORK INTERFACE (IFACE) = eth0 | eth1 | ppp0 (network adapter)

PROTOCOL (PROT) = tcp | icmp | udp …..

STATE OF THE CONNECTION (STATE) = NEW | ESTABLISHED | RELATED …..

BASED ON THE RULES THERE IS ONE ACTION

ACTION ON THE PACKET = DROP | ACCEPT | REJECT | DNAT | SNAT …..

Firewall: examples

Assume eth0 interface of a router to public Internet

- Block all incoming traffic

    iptables -A FORWARD -i eth0 -j DROP

Note: packets are discarded with no reply to the sender; in this way the firewall does not protect against flooding attacks and does not provide information for attacks based on "port scanning"

- Accept pck from outside if they refer to a TCP connection started within the network

    iptables -A FORWARD -i eth0 -m state --state ESTABLISHED -j ACCEPT

Note state "ESTABLISHED" allows to decide whether the connection originated from the inside or the outside; ESTABLISHED information is stored in the IPTABLES;

example 1

- Allow firewall to accept TCP packets for routing when they enter on interface eth0 from any IP address and are destined for an IP address of 192.168.1.58 that is reachable via interface eth1. The source port is in the range 1024 to 65535 and the destination port is port 80 (www/http)

```
iptables -A FORWARD -s 0/0 -i eth0 -d 192.168.1.58 -o eth1
    -p TCP --sport 1024:65535 --dport 80 -j ACCEPT
```

example 2

- allow the firewall to send ICMP echo-requests (pings) and in turn accept the expected ICMP echo-replies

```
iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
```

example 3

- accept at most 1 ping/second

```
iptables -A INPUT -p icmp --icmp-type echo-request -m limit
  --limit 1/s -i eth0 -j ACCEPT
```

- limiting the acceptance of TCP segments with the SYN bit set to no more than five per second

```
iptables -A INPUT -p tcp --syn -m limit --limit 5/s -i eth0 -j ACCEPT
```

example 4

- Allow the firewall to accept TCP packets to be routed when they enter on interface eth0 from any IP address destined for IP address of 192.168.1.58 that is reachable via interface eth1. The source port is in the range 1024 to 65535 and the destination ports are port 80 (www/http) and 443 (https).
- The return packets from 192.168.1.58 are allowed to be accepted too. Instead of stating the source and destination ports, you can simply allow packets related to established connections using the -m state and --state ESTABLISHED options.

```
iptables -A FORWARD -s 0/0 -i eth0 -d 192.168.1.58 -o eth1 -p TCP --sport
    1024:65535 -m multiport --dports 80,443 -j ACCEPT

iptables -A FORWARD -d 0/0 -o eth0 -s 192.168.1.58 -i eth1 -p TCP -m state
    --state ESTABLISHED -j ACCEPT
```

example 5

- allow DNS access from/to firewall

```
iptables -A OUTPUT -p udp -o eth0 --dport 53 --sport
   1024:65535 -j ACCEPT
iptables -A INPUT -p udp -i eth0 --sport 53 --dport
   1024:65535 -j ACCEPT
```

example 6

- *allow www & ssh access to firewall*

```
iptables -A OUTPUT -o eth0 -m state --state
   ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp -i eth0 --dport 22 --sport 1024:65535
   -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp -i eth0 --dport 80 --sport 1024:65535
   -m state --state NEW,ESTABLISHED -j ACCEPT
```

example 7

- allow firewall to access the Internet
  - enables a user on the firewall to use a Web browser to surf the Internet. HTTP traffic uses TCP port 80, and HTTPS uses port 443

```
iptables -A OUTPUT -j ACCEPT -m state --state
  NEW,ESTABLISHED,RELATED -o eth0 -p tcp -m multiport
  --dports 80,443 --sport 1024:65535
```

```
iptables -A INPUT -j ACCEPT -m state --state
  ESTABLISHED,RELATED -i eth0 -p tcp
```

example 8

- allow home Network to access firewall
  - in the example, eth1 is directly connected to a home network using IP addresses from the 192.168.1.0 network. All traffic between this network and the firewall is simplistically assumed to be trusted and allowed.
- iptables -A INPUT -j ACCEPT -p all -s 192.168.1.0/24 -i eth1
- iptables -A OUTPUT -j ACCEPT -p all -d 192.168.1.0/24 -o eth1

example 9

- allow loopback

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

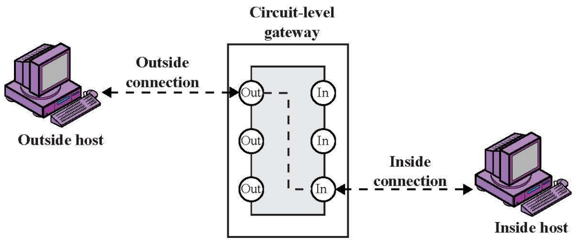
iptables administration

- a new rule immediately applies
  - no need to restart iptables
- changes are lost at reboot
  - good to insert iptables configuration in the boot sequence
- useful commands (need sudoer)
  - iptables-save > iptables.dat
  - iptables-restore < iptables.dat
- good HOWTO: https://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.txt

Firewalls: other approaches

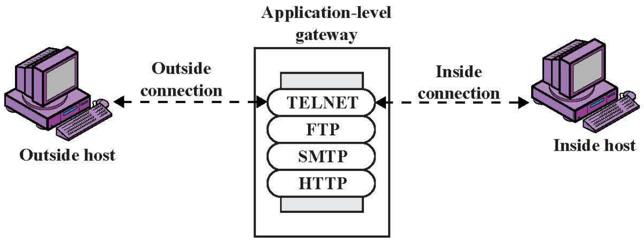
- Application level
  - use a specific application
  - fully accesses protocols
    - user requests service
    - request is accepted/denied according to defined rules
    - accepted requests are served
  - needs a proxy server for each service!
- Circuit level
  - establishes two TCP connections
  - security enforced by limiting the authorized connections

Circuit-Level Gateway



- Splices and relays two TCP connections
  - Does not examine the contents of TCP segments; less control than application-level gateway
  - checks validity of TCP connections against a table of allowed connections, before a session can be opened
  - valid session on the base of dest/src addr/ports, time of day, protocol, user and password.
  - Once session is allowed, no further checks
- Client applications must be adapted for SOCKETS
  - "Universal" interface to circuit-level gateways
- For lower overhead, application-level proxy on inbound, circuit-level on outbound (trusted users)

Application-Level Gateway



- Splices and relays application-specific connections
  - Example: Web browser proxy
  - Big overhead, but can log and audit all activity
- Can support user-to-gateway authentication
  - Log into the proxy server with username and password
- Can use filtering rules
- Need separate proxy for each application

Comparison

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

other firewalls' operations

in addition to control in/out traffic firewalls

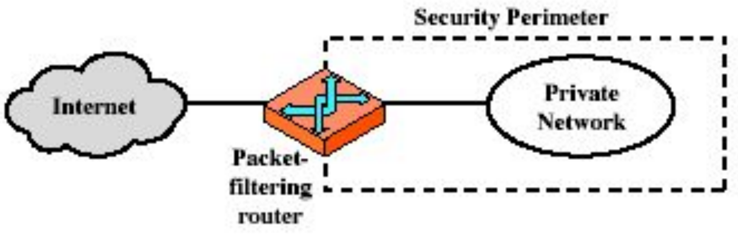
- can control band use
- hide information on internal network

Fun with Outbound

[From "The Art of Intrusion", available here]

Guess	Guess CEO's password and log into his laptop
Try	Try to download hacking tools with FTP • Oops! Personal firewall on laptop pops up a warning every time FTP tries to connect to the Internet • Kill firewall before CEO notices
Use	Use Internet Explorer object instead • Most firewalls permit Internet Explorer to connect to the Internet
Get	Get crackin'…

Firewall: where to place it



(a) Packet-filtering router

- We need servers of the network to be protected should be accessible from outside
- Solution: allow traffic for specific applications to enter (i.e., open specific ports for applications: 25 for smtp, 80 for http, …)

BUT

- Software applications can have bugs that are exploited by the attacker
- Hacker can take control of servers bypassing the firewall

Bastion Host

Bastion host is a hardened system implementing application-level gateway behind packet filter

- Trustable operating systems: run few applications and all non-essential services are turned off
- Application-specific proxies for supported services
  - Each proxy supports only a subset of application's commands, traffic is logged and audited (to analyze attacks), disk access restricted, runs as a non-privileged user in a separate directory (independent of others)
- Support for user authentication

All traffic flows through bastion host

- Packet 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

Bastion Host

- unique host that is reachable from the Internet
- massively protected host
- secure operating system (hardened or trusted)
- no unneeded software, no compilers & interpreters
- proxy server in a insulated environment (chrooting)
- read-only file system
- process checker
- integrity file system checker
- small number of services and no user accounts
- untrusted services have been removed
- saving & control of logs
- source-routing disabled

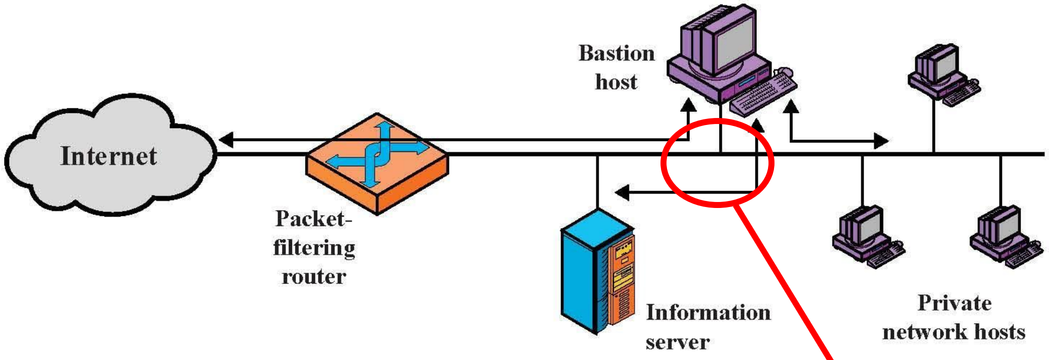
Firewall: where to place it

DeMilitarized Zone (DMZ)

- Servers that should be reachable from the outside are placed in a special area DMZ
- External connections/users can reach these servers but cannot reach the internal network because it is blocked by the Bastion host
- External connections/users that do not access these servers are dropped
- There can be several levels

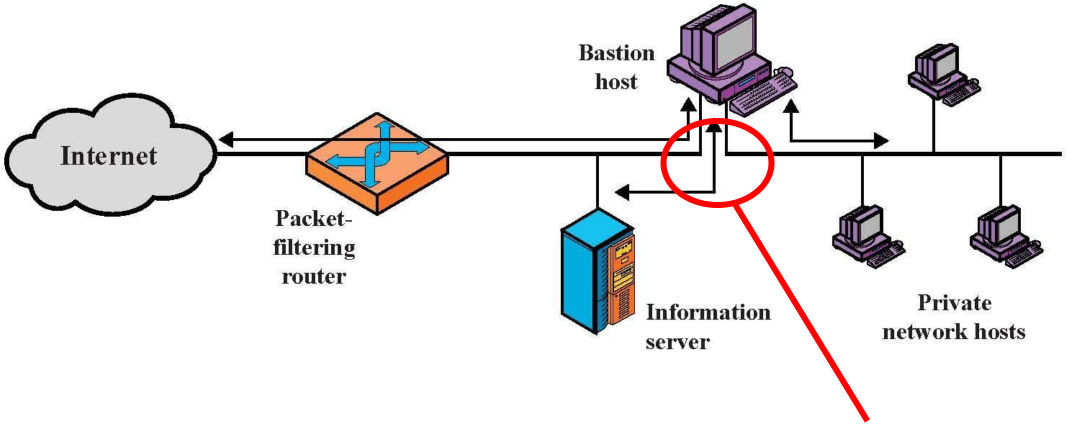
Note: great attention should be dedicated to the traffic entering the DMZ: if a hacker controls the bastion host, he can enter the internal LAN

Single-Homed Bastion Host



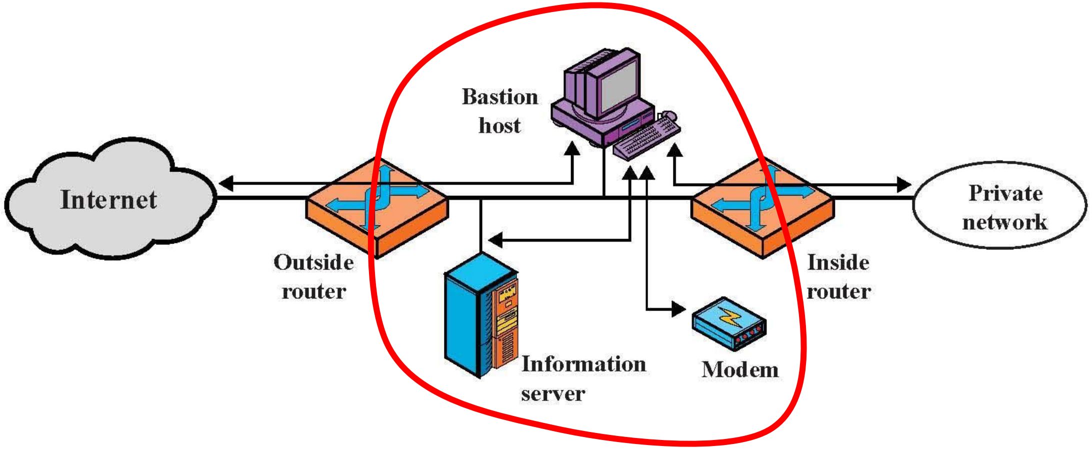
If packet filter is compromised,
traffic can flow to internal network

Dual-Homed Bastion Host



No physical connection between internal and external networks

Screened Subnet



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

Protecting Addresses and Routes

- Hide IP addresses of hosts on internal network
  - Only services that are intended to be 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
  - No need to advertise routes to internal hosts
  - Prevent attacker from advertising that the shortest route to an internal host lies through him

General Problems with Firewalls

- Interfere with networked applications
- Don't solve the real problems
  - Buggy software (think buffer overflow exploits)
  - Bad protocol design (think WEP in 802.11b)
- Generally, don't prevent denial of service
- Limited prevention of insider attacks
- Increasing complexity and potential for misconfiguration