ECS655U-ECS775P: Security Engineering

Week 8: Network Security, Firewalls

Dr Arman Khouzani

March 02, 2018

EECS, QMUL

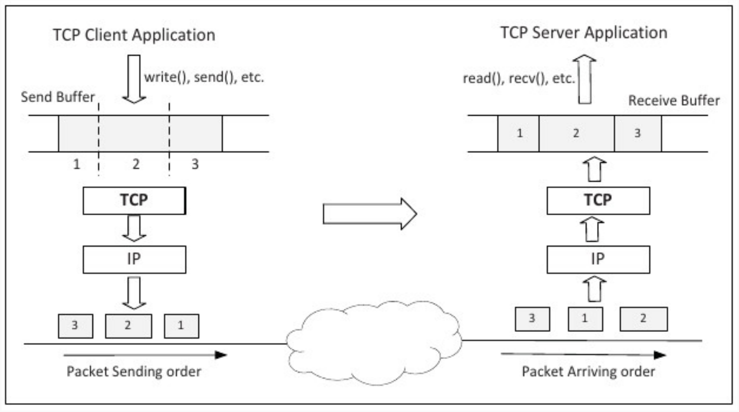
Learning Outcomes

- An Overview of computer networks
- Some network attacks and vulnerabilities
- Firewalls
    - Working out a simple packet firewall rule example

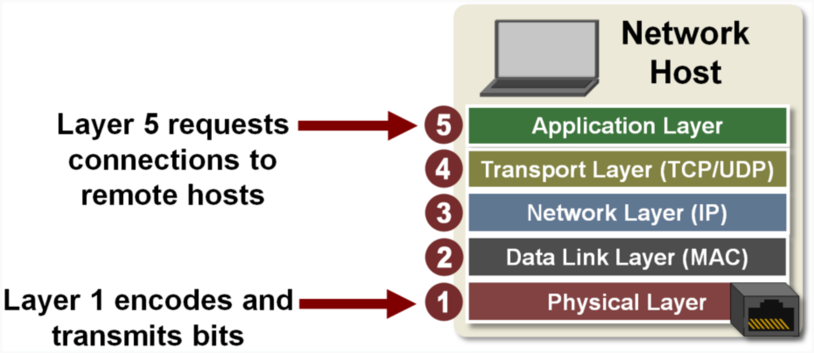
An Overview of TCP/IP

TCP/IP

- TCP/IP collection of communication protocols, originally developed by DARPA in 1980s.

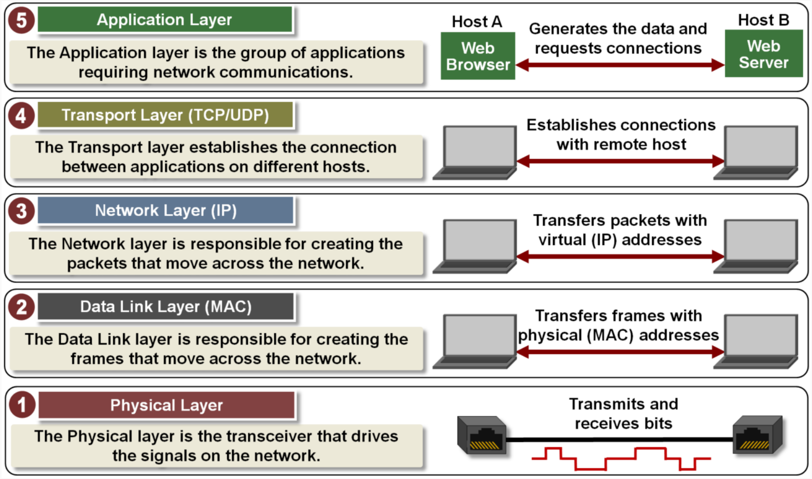


Networking Layers of TCP/IP


Network Host
Layer 5 requests connections to remote hosts
Layer 1 encodes and transmits bits
5 Application Layer
4 Transport Layer (TCP/UDP)
3 Network Layer (IP)
2 Data Link Layer (MAC)
1 Physical Layer

Ref: http://microchipdeveloper.com/tcpip:tcp-ip-five-layer-model

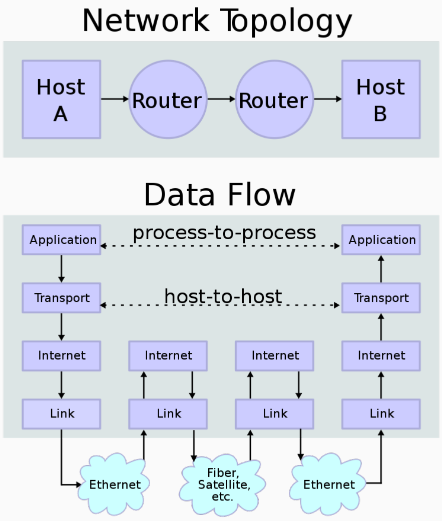
Networking Layers of TCP/IP



5 Application Layer
The Application layer is the group of applications requiring network communications.
Host A
Web Browser
Generates the data and requests connections
Host B
Web Server
4 Transport Layer (TCP/UDP)
The Transport layer establishes the connection between applications on different hosts.
Establishes connections with remote host
3 Network Layer (IP)
The Network layer is responsible for creating the packets that move across the network.
Transfers packets with virtual (IP) addresses
2 Data Link Layer (MAC)
The Data Link layer is responsible for creating the frames that move across the network.
Transfers frames with physical (MAC) addresses
1 Physical Layer
The Physical layer is the transceiver that drives the signals on the network.
Transmits and receives bits

Ref: http://microchipdeveloper.com/tcpip:tcp-ip-five-layer-model

Networking Layers



Ref: https://en.wikipedia.org/wiki/Internet_protocol_suite#Key_architectural_principles

Network Layering Model

- Each layer provides services for the layer above it, and “abstracts” the lower levels.

For example, a web browser, or a mail client (in the application layer) use the TCP layer to send requests to a remote host. In particular, they do not have to worry about “how” it does it!

- ▷ The promise of TCP is that the application data will be delivered “reliably” to the destination.
- ▷ The promise of UDP is that the application data will be delivered to its destination in the lowest latency but no guarantee.

An application chooses one based on its needs (e.g.?)

Network Layering Model

- ▷ TCP or UDP (transport layer protocols) in turn relies on the services of the IP layer: for instance, it first establishes a notion of connection between the source host (where it resides) and the TCP layer of the destination hosts, but does not have to worry about the detail of how the requests are actually routed between intermediate hops. That's IP layer's business!
- ▷ IP on the other hand relies on the link layer to do its service: for instance, it does not have to worry about the physical medium of transmission, the topology of the network, avoiding collisions, etc.

Network Layering Model

This not having to worry goes both-ways:

- IP layer does not have to worry about who are the two ends of the communication.
- TCP/UDP does not have to be concerned with the content of the data, or the detail of the application

Link Layer (a.k.a. Data Link Layer)

- **Link layer**: includes the protocol necessary to send information between two hosts that are connected immediately to the same media (for example, a coaxial cable or over the air via radio waves).
- ▷ It also includes the mechanisms for replication over several links by “switches” (a.k.a. “bridges”).
- Example protocols:
  - ▷ IEEE 802.3 and Ethernet (most prevalent)
  - ▷ FDDI and Optical Fibre
  - ▷ IEEE 802.11 Wireless LAN, IEEE 802.16 WiMAX

- At link layer, the message unit is called a *frame*.

Internet Layer (a.k.a. Network Layer)

- ► Not all devices can be immediately connected to each other! so there is a need to transmit data across multiple devices, hence the **Internet Layer**
- ▷ Each host is assigned an Internet Protocol (IP) address, each “nodes” transmits the data to the next best node to reach the destination IP address (“routing”).
  - At the Internet layer, the message unit is called a *packet*.

Transport Layer

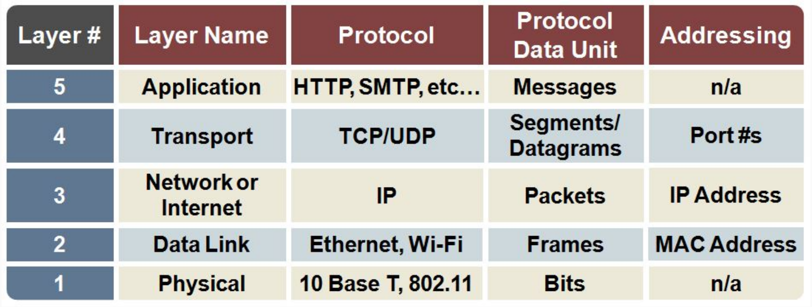
- How can the sending host know whether its data successfully made it to the destination? This (+flow control!) is the main job of the **transport layer**.
  - For example, it detects whether any part of the information is delivered out of order (how?) or went missing (how?) or any part is delivered with errors (again, how?)
  - and in such cases, it rectifies the situation (how?)
- Question: a host maybe running multiple applications requiring connection (e.g. web browser, ftp transfer, email, VoIP, etc), how is each data delivered to the right application?

- At this layer, the message unit is called a *segment*.

Application Layer

- ► e.g.: Simple Message Transfer Protocol (SMTP) is used to handle sending of emails, the Hypertext Transfer Protocol (HTTP) is used to request a web page within a browser, File Transfer Protocol for file transfer, etc.
- ▷ The protocols are not concerned with how the information will reach the destination, but only work on defining the content of the information being transmitted and the logic of interactions.

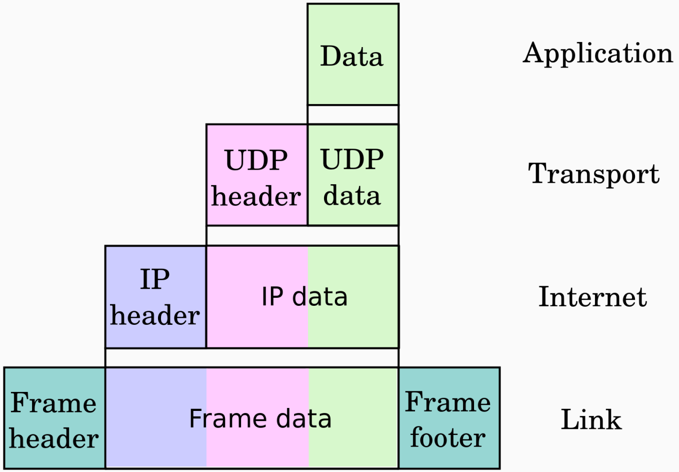
Layer Terminology



Layer #	Layer Name	Protocol	Protocol Data Unit	Addressing
5	Application	HTTP, SMTP, etc…	Messages	n/a
4	Transport	TCP/UDP	Segments/ Datagrams	Port #s
3	Network or Internet	IP	Packets	IP Address
2	Data Link	Ethernet, Wi-Fi	Frames	MAC Address
1	Physical	10 Base T, 802.11	Bits	n/a

Ref: http://microchipdeveloper.com/tcpip:tcp-ip-five-layer-model

Encapsulation

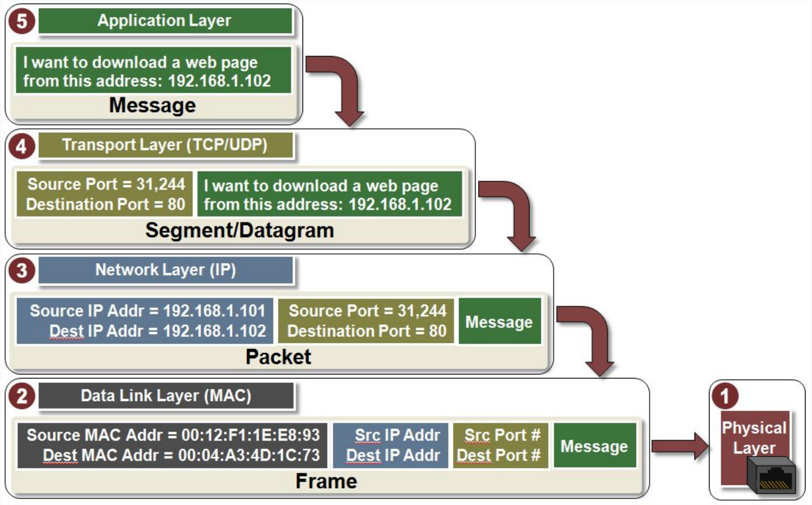


Ref: https://en.wikipedia.org/wiki/Internet_protocol_suite#Key_architectural_principles

Encapsulation

- Example items in the **TCP** header:
  - Source port (why?)
  - Destination port (why?)
  - “Checksum” (why?)
  - Sequence number (why?)
- Example items in a **IP** header:
  - Destination IP address (why?)
  - Source IP address (why?)
- Example items in a **Link** layer protocol header:
  - Destination MAC address **Note: Destination here means the next immediate node!**
  - Source MAC address **Note: Source here means the current node!**

Encapsulation: Example

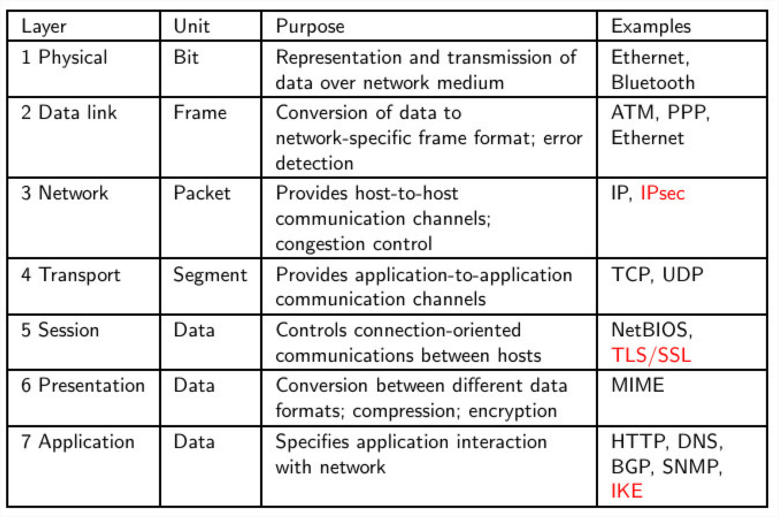


5
Application Layer
I want to download a web page from this address: 192.168.1.102
Message
4
Transport Layer (TCP/UDP)
Source Port = 31,244
Destination Port = 80
I want to download a web page from this address: 192.168.1.102
Segment/Datagram
3
Network Layer (IP)
Source IP Addr = 192.168.1.101
Dest IP Addr = 192.168.1.102
Source Port = 31,244
Destination Port = 80
Message
Packet
2
Data Link Layer (MAC)
Source MAC Addr = 00:12:F1:1E:E8:93
Dest MAC Addr = 00:04:A3:4D:1C:73
Src IP Addr
Dest IP Addr
Src Port #
Dest Port #
Message
Frame
1
Physical Layer

Ref: http://microchipdeveloper.com/tcpip:tcp-ip-five-layer-model

TCP/IP

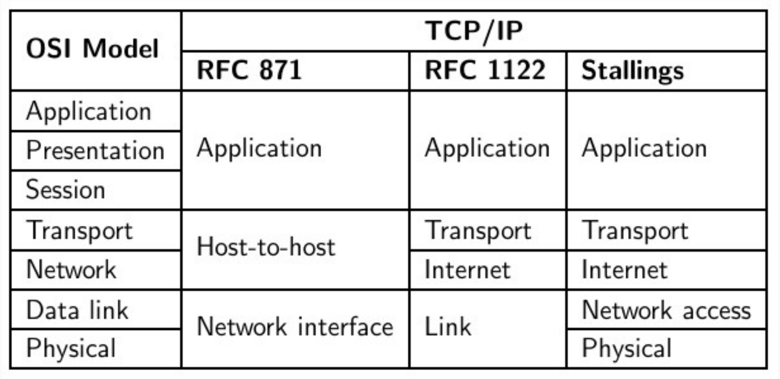
The Open System Interconnection Standard (OSI) Model:



Layer	Unit	Purpose	Examples
1 Physical	Bit	Representation and transmission of data over network medium	Ethernet, Bluetooth
2 Data link	Frame	Conversion of data to network-specific frame format; error detection	ATM, PPP, Ethernet
3 Network	Packet	Provides host-to-host communication channels; congestion control	IP, IPsec
4 Transport	Segment	Provides application-to-application communication channels	TCP, UDP
5 Session	Data	Controls connection-oriented communications between hosts	NetBIOS, TLS/SSL
6 Presentation	Data	Conversion between different data formats; compression; encryption	MIME
7 Application	Data	Specifies application interaction with network	HTTP, DNS, BGP, SNMP, IKE

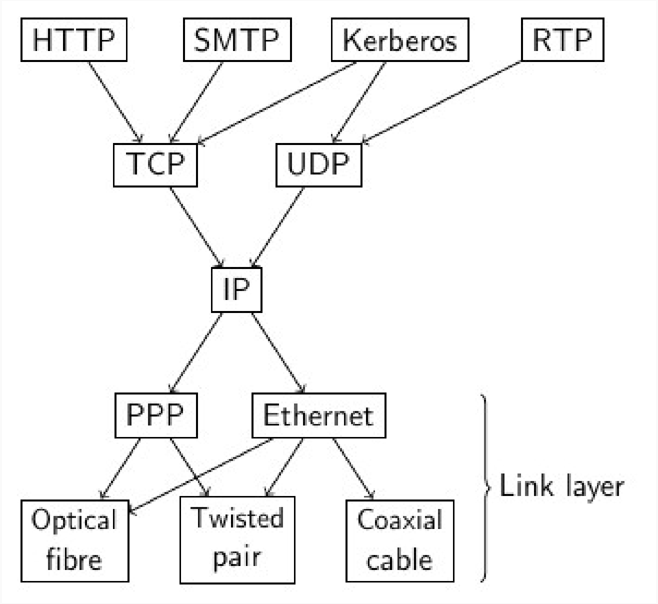
TCP/IP

OSI model vs TCP/IP:



OSI Model	TCP/IP		
	RFC 871	RFC 1122	Stallings
Application	Application	Application	Application
Presentation			
Session			
Transport	Host-to-host	Transport	Transport
Network		Internet	Internet
Data link	Network interface	Link	Network access
Physical			Physical

TCP/IP


HTTP
SMTP
Kerberos
RTP
TCP
UDP
IP
PPP
Ethernet
Optical fibre
Twisted pair
Coaxial cable
Link layer

TCP/IP

Different Processing Orders Provide Different Services.
Things you should know:

- ▷ *HTTP* → *TCP* → *IP* → *Ethernet*: represents the usual order of processing for HTTP messages
- ▷ *HTTP* → *SSL*/*TLS* → *TCP* → *IP* → *Ethernet*: An HTTP message is encapsulated in an SSL/TLS PDU to provide Confidentiality/Integrity/Authentication;

TCP/IP

Other possibilities (you don't need to know!):

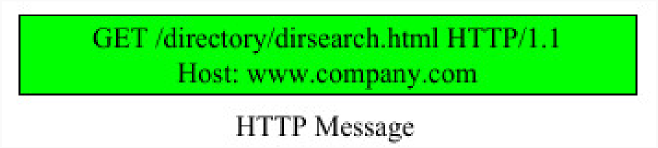
- ▷ *HTTP* $\rightarrow$ *TCP* $\rightarrow$ *IP* $\rightarrow$ *IP* $\rightarrow$ *Ethernet*: An IP datag. can be enc. in an IP datagram, (IP tunnelling);
- ▷ *HTTP* $\rightarrow$ *TCP* $\rightarrow$ *IP* $\rightarrow$ *IPsec* $\rightarrow$ *Ethernet* IPsec transport mode (IP datagram is processed by IPsec);
- ▷ *HTTP* $\rightarrow$ *TCP* $\rightarrow$ *IP* $\rightarrow$ *IP* $\rightarrow$ *IPsec* $\rightarrow$ *Ethernet*: IPsec tunnel mode, where an IP datagram is enc. in an IP datagram and then processed by IPsec.

TCP/IP: an example

- ▷ Web server and web browser communicate using an application layer protocol.
- ▷ In this case, the protocol is Hypertext Transfer Protocol (HTTP).
- ▷ Users invoke applications which “speak” using application protocol.
- ▷ Applications interact with a transport protocol to send or receive data.

TCP/IP: an example

HTTP outline:
GET /directory/dirsearch.html HTTP/1.1
Host: www.company.com.co.uk


GET /directory/dirsearch.html HTTP/1.1
Host: www.company.com

HTTP Message

TCP/IP: an example

Transport layer provides end-to-end communication service between pairs of applications.

- the transport layer protocol selected here by our applications is the Transport Control Protocol (TCP):
  - A reliable, connection-oriented transport protocol.
  - Divides stream of application messages into segments.
- Transport layer interacts with Internet Layer to send or receive data.
- In general, a transport protocol may be
  - reliable or unreliable,
  - connection-oriented or connectionless,
  - and flow may or may not be regulated.
- Other transport layer protocols: UDP, ICMP, SCTP…

TCP/IP: an example

Transport layer provides end-to-end communication service between pairs of applications.

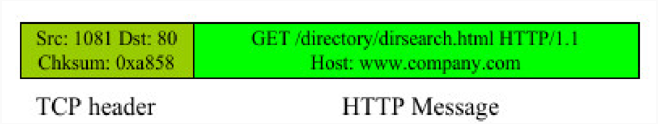
- the transport layer protocol selected here by our applications is the Transport Control Protocol (TCP):
  - A reliable, connection-oriented transport protocol.
  - Divides stream of application messages into segments.
- Transport layer interacts with Internet Layer to send or receive data.
- In general, a transport protocol may be
  - reliable or unreliable,
  - connection-oriented or connectionless,
  - and flow may or may not be regulated.
- Other transport layer protocols: UDP, ICMP, SCTP…

TCP/IP: an example

TCP outline:

- ▷ Source Port: 1081
- ▷ Destination Port: 80
- ▷ Checksum: 0xa858

- Ports allow identification of applications.
- Checksum for error detection.



TCP/IP: an example

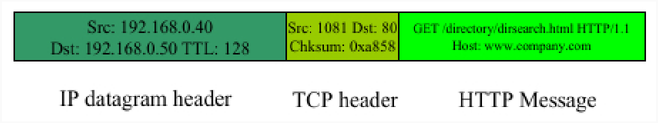
Internet Layer: Responsible for routing communications between one machine and another.

- Accepts requests to send transport layer messages to destination address.
- Internet Protocol (IP) encapsulates messages in IP datagram with IP header and uses routing algorithm to decide whether to send directly or indirectly.
- Also handles incoming IP datagrams.
  - If addressed to local machine, remove the IP datagram header and pass up to transport layer.

TCP/IP: an example

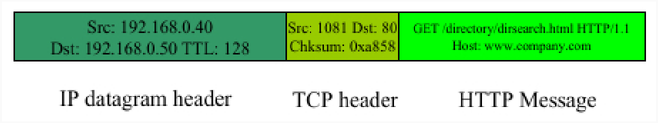
IP outline:

- Time to live: 128
- Header checksum: 0x57d1
- Source: pelican (192.168.0.40)
- Destination: www.company.co.uk (192.168.0.50)



TCP/IP: an example

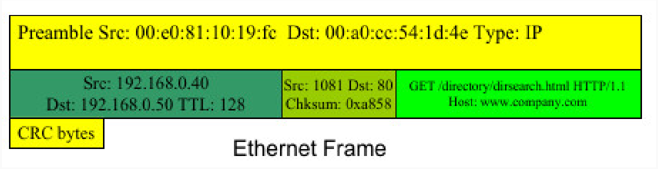
- ▷ Time to live: 128
- ▷ Header checksum: 0x57d1
- ▷ Source: pelican (192.168.0.40)
- ▷ Destination: www.company.co.uk (192.168.0.50)



TCP/IP: an example

In this example, say Ethernet is used.

- Ethernet outline:
  - Synchronisation preamble.
  - Destination: 00:a0:cc:54:1d:4e.
  - Source: 00:e0:81:10:19:fc.
  - Type: IP.
  - Data in frame.
  - Cyclic Redundancy Check (CRC).



Preamble Src: 00:e0:81:10:19:fc Dst: 00:a0:cc:54:1d:4e Type: IP
Src: 192.168.0.40 Dst: 192.168.0.50 TTL: 128
Src: 1081 Dst: 80 Chksum: 0xa858
GET /directory/dirsearch.html HTTP/1.1 Host: www.company.com
CRC bytes
Ethernet Frame

TCP/IP: an example

- ▷ The network interface card of all the machines that are on the same Ethernet wire (including the router/switch) keep inspecting the header of all the ethernet data-frames, and if the ethernet address, i.e., the destination MAC (Media Access Control) address of a data-frame matches theirs, they will pick it up for processing;

TCP/IP: an example

- A router looks at (a prefix of the) IP address of the destination, then based on its “routing table” and routing rules, decides what is the next immediate link, so it replaces the source ethernet address with its own ethernet address (which was the destination ethernet address of previous step), and replaces the destination ethernet address (the next hop router);

TCP/IP: an example

- This process goes on till the packet reaches the machine with destination IP address. At that point, the IP datagrams, the TCP messages and eventually, the HTTP message is reconstructed at the IP, TCP and the application layer of the destination machine, respectively.

Example Network Attacks

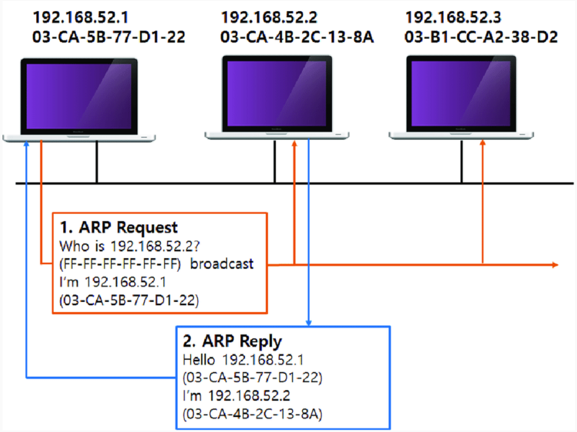
ARP Spoofing

What is **ARP**?

- ▷ the IP Layer isn't sitting "listening to the wire" in a LAN
- ▷ The network interface card (NIC) can be identified using the MAC address, but how does a device know which MAC address to send the packet to, given that its TCP/IP stack has just passed us the destination IP address?
- ▶ Answer: **Address Resolution Protocol (ARP)**

ARP Spoofing

ARP: A network protocol used to find the hardware (MAC) address of a host from an IP address (on a LAN)



ARP request-response protocol example.

ARP Spoofing

ARP spoofing/Poisoning: there is no DOA for responses – so false ARP responses could be sent by an adversary!

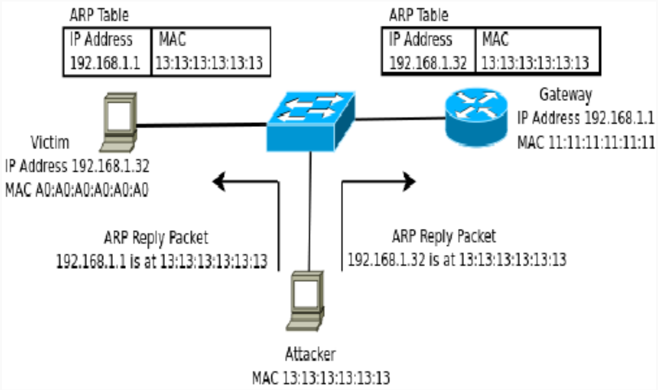
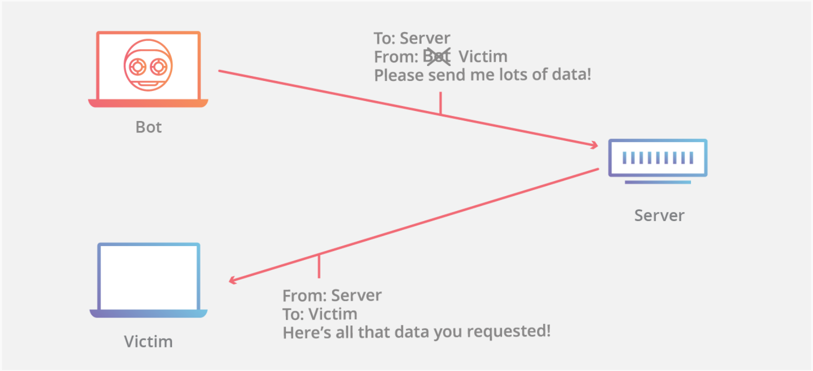


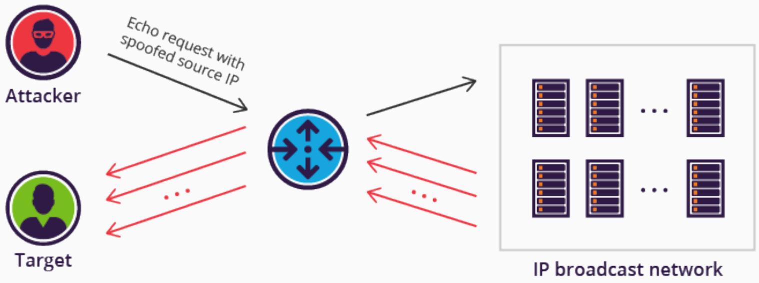
Illustration of ARP cache poisoning attack.

IP Address Spoofing Attacks



A DDoS attack enabled by IP Spoofing.

IP Address Spoofing Attacks

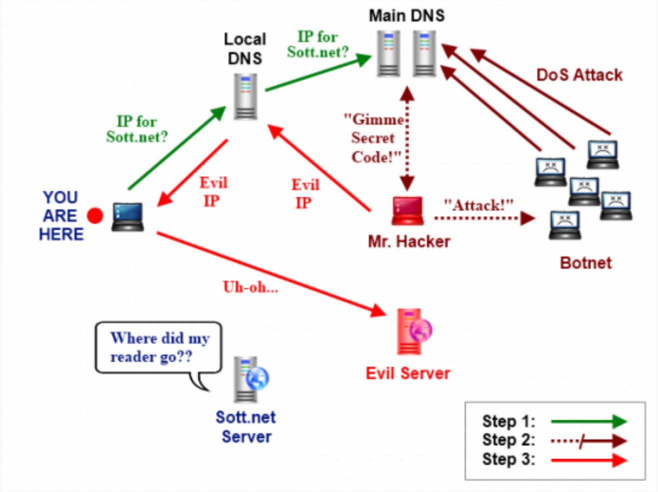


Smurf Attack enabled by IP spoofing. Read more about it here.

DNS Spoofing

- ▶ Domain Name System (DNS) is a system that associates domain names (e.g. URLs, email addresses and other human-readable domain names) with their corresponding IP addresses.
- ▷ In a DNS spoofing attack, attacker spoofs DNS responses in order to reroute a specific domain name to a different IP address (e.g., one that they control)

DNS Spoofing



An example scenario of a DNS cache poisoning attack (ref)

Firewalls

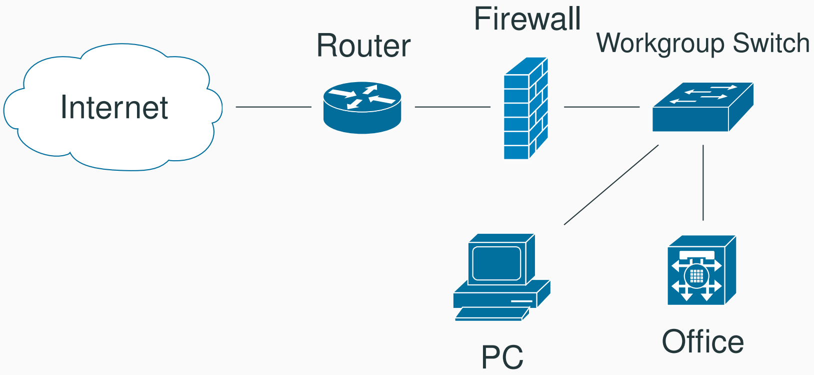
Firewalls

- Firewall:
  - A component or a set of components that restricts access between a protected network and the Internet or between different sections of a network.
- Packet Filtering:
  - The action a device takes to selectively control the flow of data to and from a network
  - Packet filtering is an important technique to implement access control on the subnetwork-level for packet oriented networks, e.g. the Internet
  - A synonym for packet filtering is *screening*

Firewalls

- ▷ The role of firewall is insulating the internal systems from external networks while providing some controlled connectivity. (If no connectivity is needed, then disconnect them physically, or “air-gap” them!).
- ▷ The traffic going in each direction must pass through the firewall barrier. A firewall security policy dictates which traffic is authorized to pass in each direction.
- ▷ A firewall may be designed to operate as a filter at the level of IP packets, in which case it is also called a packet filter, or may operate at a higher protocol layer.

Firewalls



▷ The firewall is inserted between the premises network and the Internet to establish a controlled link and to erect an outer security “perimeter”.

Firewalls

- The aim of this perimeter is to protect the premises network from attacks originating from the Internet and to provide a single choke point where security and auditing can be imposed.
- The level of “trust” is different between the internal network and the “Internet”.

Firewalls

What firewalls can do:

- A firewall can enforce a security policy, i.e. concerning access control
- it can log Internet activity efficiently
- it can block unwanted traffic if the traffic can be characterized,
  - e.g. with an IP 5-tuple: IP source address, IP destination address, source port number, destination port number, transport protocol
- it can limit exposure to security problems in one part of a network, i.e., contain a breach.

Firewalls

What firewalls can do:

- A firewall can enforce a security policy, i.e. concerning access control
- it can log Internet activity efficiently
- it can block unwanted traffic if the traffic can be characterized,
  - e.g. with an IP 5-tuple: IP source address, IP destination address, source port number, destination port number, transport protocol
- when used between different internal subnetworks, it can limit exposure to security problems in one part of a network, i.e., contain a breach (known as “network compartmentalisation”).

Firewalls

What firewalls can NOT do:

- firewalls cannot protect against “malicious insiders”;
- they cannot protect against connections that do not go through it;
- A firewall cannot fully protect against viruses,
  - e.g. viruses can still spread through emails (as usual), and the email service is allowed by the firewall;
- they cannot guarantee that the parameters that their rules are based are not spoofed by attackers, as they normally do not perform cryptographic operations, e.g. message authentication;
- they cannot set themselves up correctly (rules can become large & complicated for manual checking)

Firewalls

Two types of packet filtering:

- Stateless: each decision (allow/drop/reject/log) is made based on inspecting a single packet, e.g.: drop any packet with a specific source ip-address.
- Stateful: decisions may depend on previous packets (to take into account the state of the connection/operations that the packet is part of).
  - Stateful firewalls with application inspection maintain a table of open connections, inspecting the payload of some packets and intelligently associating new connection requests with existing legitimate ones.
- Stateful firewalls are more flexible than stateless, but require more resources, & may introduce delay.

Firewalls

Specifying packet filtering rules:

- As a packet filter protects one part of a network from another with different trust levels, there is a notion of the direction of traffic:
  - *Inbound:* The traffic is coming from an interface which is outside the protected network and its destination can be reached on an interface which is connected to the protected network
  - *Outbound:* the opposite of inbound
  - For every packet filtering rule this direction is specified as either *inbound*, *outbound*, or *either*.

Firewalls

Specifying packet filtering rules:

- Source and destination address specifications can make use of wildcards, e.g. 125.26.*.* denotes all addresses starting with 125.26.
- For source and destination ports we sometimes write ranges, e.g. $>$ 1023.
- Filtering rules are usually applied in the order of specification, that means the first rule that matches a packet is applied.

Firewalls: an example

Let us consider setting up a packet filtering firewall ruleset for setting up a mail server (borrowed from *these slides*):

- ▷ we should specify that incoming and outgoing email should be the only allowed traffic into and out of a protected network;
- ▷ Email is relayed between two servers by transferring it to an SMTP daemon on the target server;
- ▷ For SMTP, the server port is 25, and client port is some number greater than 1023.

First, we create a ruleset that specifies that the only allowed traffic should be incoming and outgoing emails into and out of a protected network.

Firewalls: example

Rule	Direc.	Src. Add.	Dest. Add.	Proto.	Src. Prt	Dst. Prt	ACK	Action
A	In	Ext.	Int.	TCP		25		Permit
B	Out	Int.	Ext.	TCP		> 1023		Permit
C	Out	Int.	Ext.	TCP		25		Permit
D	In	Ext.	Int.	TCP		> 1023		Permit
E	Either	Any	Any	Any		Any		Deny

- ▷ Rule A allows incoming email to enter the network and rule B allows the acknowledgements to exit the network;
- ▷ Rules C and D are analogous for outgoing email;
- ▷ Rule E denies all other traffic.

Firewalls: example

Rule	Direc.	Src. Add.	Dest. Add.	Proto.	Src. Prt	Dst. Prt	ACK	Action
A	In	Ext.	Int.	TCP		25		Permit
B	Out	Int.	Ext.	TCP		> 1023		Permit
C	Out	Int.	Ext.	TCP		25		Permit
D	In	Ext.	Int.	TCP		> 1023		Permit
E	Either	Any	Any	Any		Any		Deny

- Consider a packet entering the protected subnet that has a spoofed IP source address from the internal network:
  - As all allowed inbound packets must have external source and internal destination addresses (Rules A and D) this packet is successfully blocked
  - The same holds for outbound packets with external source addresses (Rules B and C)

Firewalls: example

Rule	Direc.	Src. Add.	Dest. Add.	Proto.	Src. Prt	Dst. Prt	ACK	Action
A	In	Ext.	Int.	TCP		25		Permit
B	Out	Int.	Ext.	TCP		> 1023		Permit
C	Out	Int.	Ext.	TCP		25		Permit
D	In	Ext.	Int.	TCP		> 1023		Permit
E	Either	Any	Any	Any		Any		Deny

- Consider now telnet traffic:
  - As a telnet server resides usually at port 23, and all allowed inbound traffic must be either to port 25 or to a port number > 1023, incoming packets to initiate an incoming telnet connection are successfully blocked
  - The same holds for outgoing telnet connections

Firewalls: example

Rule	Direc.	Src. Add.	Dest. Add.	Proto.	Src. Prt	Dst. Prt	ACK	Action
A	In	Ext.	Int.	TCP		25		Permit
B	Out	Int.	Ext.	TCP		> 1023		Permit
C	Out	Int.	Ext.	TCP		25		Permit
D	In	Ext.	Int.	TCP		> 1023		Permit
E	Either	Any	Any	Any		Any		Deny

- However, the ruleset is flawed as, for e.g., it does not block the X11-protocol for remote operation of X-Windows:
  - An X11-server usually listens at port 6000, clients use port numbers > 1023
  - Thus, an incoming X11-request is not blocked (rule D), neither is any X11-acknowledgements (rule B)
  - X11-protocol offers many vulnerabilities to attackers: e.g. reading & manipulating the display & keystrokes

Firewalls: example

Rule	Direc.	Src. Add.	Dest. Add.	Proto.	Src. Prt	Dst. Prt	ACK	Action
A	In	Ext.	Int.	TCP	> 1023	25		Permit
B	Out	Int.	Ext.	TCP	25	> 1023		Permit
C	Out	Int.	Ext.	TCP	> 1023	25		Permit
D	In	Ext.	Int.	TCP	25	> 1023		Permit
E	Either	Any	Any	Any	Any	Any		Deny

This is fixed by including the source ports into the specification:

- ▷ As now outbound traffic to ports > 1023 is allowed only if the source port is 25 (Rule B), traffic from internal X-clients or X-servers (port > 1023) will be blocked
- ▷ Same holds for inbound traffic to ports > 1023 (Rule D)

Firewalls: example

Rule	Direc.	Src. Add.	Dest. Add.	Proto.	Src. Prt	Dst. Prt	ACK	Action
A	In	Ext.	Int.	TCP	> 1023	25		Permit
B	Out	Int.	Ext.	TCP	25	> 1023		Permit
C	Out	Int.	Ext.	TCP	> 1023	25		Permit
D	In	Ext.	Int.	TCP	25	> 1023		Permit
E	Either	Any	Any	Any	Any	Any		Deny

- However, it can not be assumed for sure that an attacker will not use port 25 for his attacking X-client:
  - In such a case the above filter will let the traffic pass

Firewalls: example

Rule	Direc.	Src. Add.	Dest. Add.	Proto.	Src. Prt	Dst. Prt	ACK	Action
A	In	Ext.	Int.	TCP	> 1023	25	Any	Permit
B	Out	Int.	Ext.	TCP	25	> 1023	Yes	Permit
C	Out	Int.	Ext.	TCP	> 1023	25	Any	Permit
D	In	Ext.	Int.	TCP	25	> 1023	Yes	Permit
E	Either	Any	Any	Any	Any	Any	Any	Deny

This can be solved by specifying TCP's ACK-flag:

- ▷ it is now disallowed (by Rule B) to initiate a new TCP connection in the outbound direction to ports > 1023, as TCP's connect-request does not have the ACK-flag set
- ▷ The same holds for the inbound direction, as rule D requires the ACK-flag to be set

Firewalls: example

Rule	Direc.	Src. Add.	Dest. Add.	Proto.	Src. Prt	Dst. Prt	ACK	Action
A	In	Ext.	Int.	TCP	> 1023	25	Any	Permit
B	Out	Int.	Ext.	TCP	25	> 1023	Yes	Permit
C	Out	Int.	Ext.	TCP	> 1023	25	Any	Permit
D	In	Ext.	Int.	TCP	25	> 1023	Yes	Permit
E	Either	Any	Any	Any	Any	Any	Any	Deny

▷ As a basic rule, any filtering rule that permits incoming TCP packets for outgoing connections should require the ACK-flag to be set.

Firewalls: DMZ

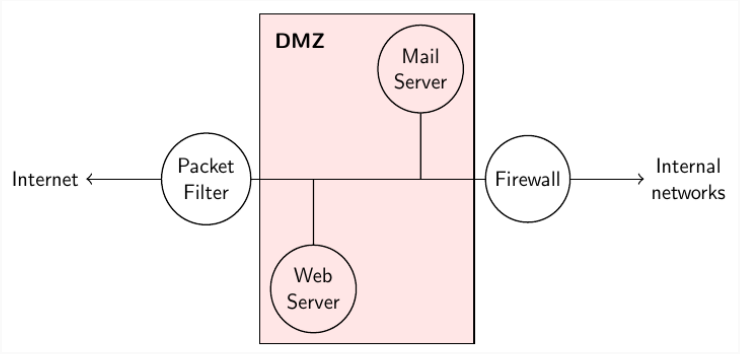
- Typically, an institution has publicly accessible services, like web-servers and mail-servers, and at the same time, their employee and internal machines need to have access to the internet.
- So we need to design our networks so that:
  - we can provide the desired connectivity to public while minimizing the exposure of our internal machines to malicous behaviour
  - we can minimize and contain the damage that occurs if one of our machines is compromised
- The second requirement suggests that our web servers and mail servers should be isolated from other machines

Firewalls: DMZ

- We should try to isolate servers that are required to be “Internet-facing” (such as mail, web and DNS servers)
  - They should be not be fully exposed to the Internet and they should be isolated from internal networks
- The solution is to create a **de-militarized zone (DMZ)**:
  - Access to a DMZ from the Internet is restricted (to protect machines within the DMZ from the Internet)
  - Access from the DMZ to an internal network is restricted (to protect machines within the internal networks from the DMZ)

Firewalls: DMZ

A DMZ is typically implemented using firewalls:



Firewalls: DMZ

- The Internet-DMZ packet filter restricts traffic to the DMZ and the internal network.
    - The rule set would (typically) allow all TCP packets to the mail server on port 25 and deny all other TCP traffic to the mail server
    - allow all TCP packets to the web server on port 80 and deny all other TCP traffic to the web server
- The firewall between the DMZ and the internal network would drop malicious traffic generated by a compromised web or mail server.

Questions?