6CCS3NSE Network Security

L1 Basics of Network Security

Security is a security policy plus a mechanism to implement it.

Threat Model: the answers to the key security questions we have to ask

- Who/what is being protected?
- Who/what is attacking?
- What are their powers?

Policy: codifies “desired: behaviour. It contains the:

- Principals (actors and participants, they may be generalised as roles, such as “student” or “lecturer”)
- Actions permitted/disallowed to principals on objects being protected.
  E.g: Only “root” can execute this script, IP packet has been sent by the host in src field, Voter has voted at most once in election.

Security Properties

The type of policy depends on which properties we want to achieve.

- **Confidentiality:** protection of content (information) from unauthorised parties
- **Integrity:** protection of content from modification by unauthorised parties
- **Availability:** prevent deliberate overload; keep system/resource usable by legitimate users

Mechanism: the means of enforcing a policy.

- Only “root” can execute this script -> Use password for root + check if user id is root.

Types of Mechanisms:

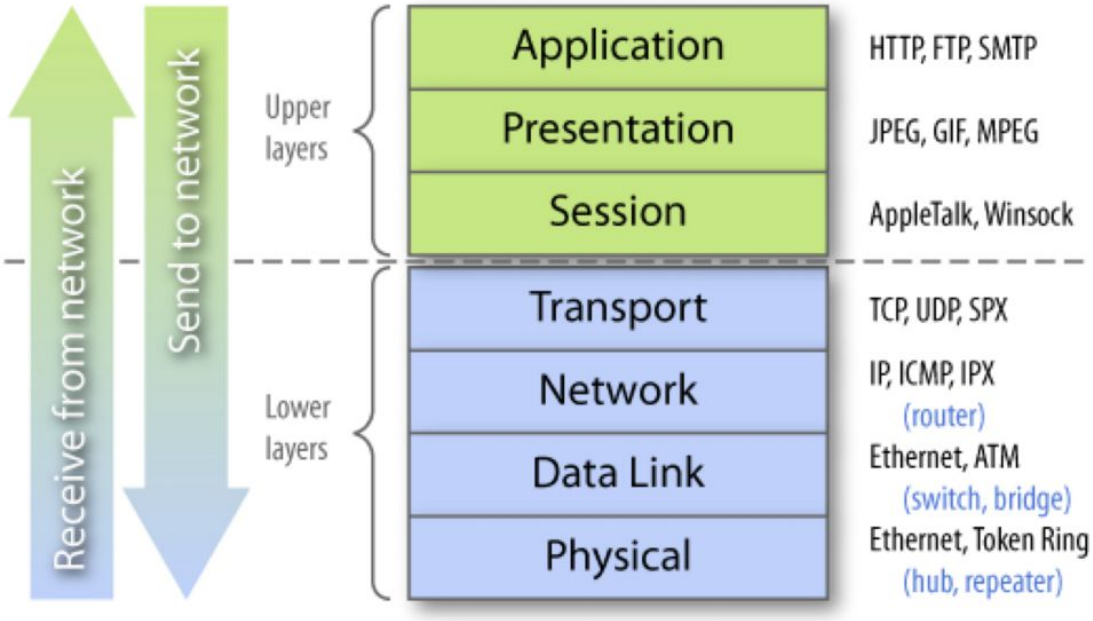
- Deter = to make it too difficult or not worthwhile to attack.
- Detect = monitor for attacks
- Deny = prevent unauthorised access
- Delay = slow down users (more suited for physical security)
- Defend = Take remedial steps after attack
- (Insure) = Pass consequences of risk to someone else!
  Example: protecting against viruses in PCs.

Security realised through Policy + Mechanism Policy codifies “desired” behaviour. Mechanism is the means of enforcing policy.

Mechanisms may involve authentication, authorisation, physical protection, cryptography, Economics (if we have to pay to send email, we might not have spam), Deception (honeypots: putting a dummy server with minimal functionality so it gets hacked. You then get the attacker to think he is attacking a server to observe how he is behaving and tailor how they work), randomness and unpredictability.

OSI Model

The standard model for the internet.



PLEASE DO NOT TALK -> Lower layers
SPAstic -> Upper layers

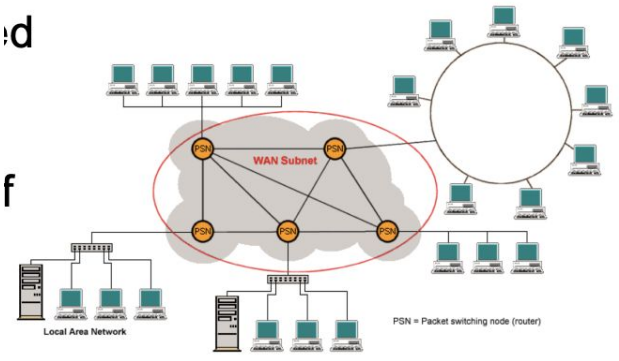
- **Physical:** Wire for wired network or the waves in wireless network.
    - *I physically connect something.*
- **Data Link:** Forwarding packets within a network. Understanding who is talking to you or who is sending information to you. We are changing a broadcast medium to a unicast medium.
    - *I link myself to the conversation by broadcasting my voice.*
- **Network:** Routing packets through different networks so devices which are not wired together can communicate (addressing nodes, selecting paths).
    - *I address someone who I want to speak to directly*
- **Transport:** provides end-to-end connectivity, correcting errors, lost packets, and potentially reordering packets to maintain relative order of packets.
    - *I put a pipe between myself and the person I want to talk to so that I make sure he can hear me*

Link Layer

The link layer is a broadcast medium (whether its wired or wireless). It is easy to access and easy to tamper with.

Network Layer

1. The network layer is needed to route between different LANs.
2. The Internet is a network of networks.
3. Info needs to be shared to find routes, but network operators want to keep private data private!



Transport Layer

It's an end-to-end pipe for bits, from sender to receiver. A way for communication between two devices. This pipe adds some capabilities such as:

- Reliability – correct for errors.

- Ordering of data (Typically FIFO – data is delivered in the order it was sent to receiver) [in the layers above we do not mind about the order, we just care about if it had been sent.]
- Dealing with network effects such as congestion.

In the OSI model
Each layer and protocol exposes **information** and **functionality.** These layers are connected and depending on what you do at layer 2 may compromise what you can do in layer 3. Sophisticated attacks exploit multiple layers.

Types of Attack

Jamming: affects the availability so that legitimate packets take too much to arrive. We jam at the link layers. *It's like if I start screaming really loud in a room so nobody can communicate.*

- Jamming at the network or transport layer is what we call DDoS attacks. In this case we are not in the LAN but we send so many requests that it is not possible for a service to answer.

Portable jammer

Wifi
3G/4G
5-20 meters

£100-200

Sniffing: involve listening to network communications which are not intended for you.

- Network cards filter all the packets that destined for you. But card may be put in a promiscuous mode to get all packets from network because physically you are getting all that information anyway.
- Sniffing can be used to debug network and get diagnostics of it. It can sometimes have a positive purpose.

Spoofing: pretending to be somebody you are not; masquerading.

- We can be someone else on the internet by just changing the **IP address.**
- In our local area networks we can change who we are by changing the **MAC address**.
    - You can change your MAC address in an airport to get free internet again. As our mac address is not registered and therefore they give you a new connection.

Spoofing forms the basis of a lot of attacks because it is very easy to change your address.

Confidentiality	sniffing	*encrypt*
Integrity	spoofing	*message digest + sign*
Availability	jamming	*account and police*

BUT: These solutions may not always be practical!

Confidentiality	sniffing
Integrity	spoofing
Availability	jamming

Hijacking: taking over a network resource which belongs to someone else.
Poisoning: is corrupting a store of information (e.g., a cache) that is relied upon for future communications.

L2 Sniffing

Networking

What happens under the hood when we go to google:

1. Browser establishes makes a DNS with the URL it wants to retrieve an ip address.
2. It then establishes a TCP (TCP allows us to monitor what has been sent or not) connection with www.google.com server, and then
3. It asks using HTTP (command GET) for the index web page (the one you put the text to search for) that it displays (render) in the browser.

IP address *-> address in the building*
Port *-> office in the building*

Sending a package in LAN

Given that we are trying to send something through our LAN we could go through the following steps of encapsulation to send a request.

Encapsulating: getting the package ready to be sent over a specific channel.
Demultiplexing: removing layers of the query to retrieve the original query.

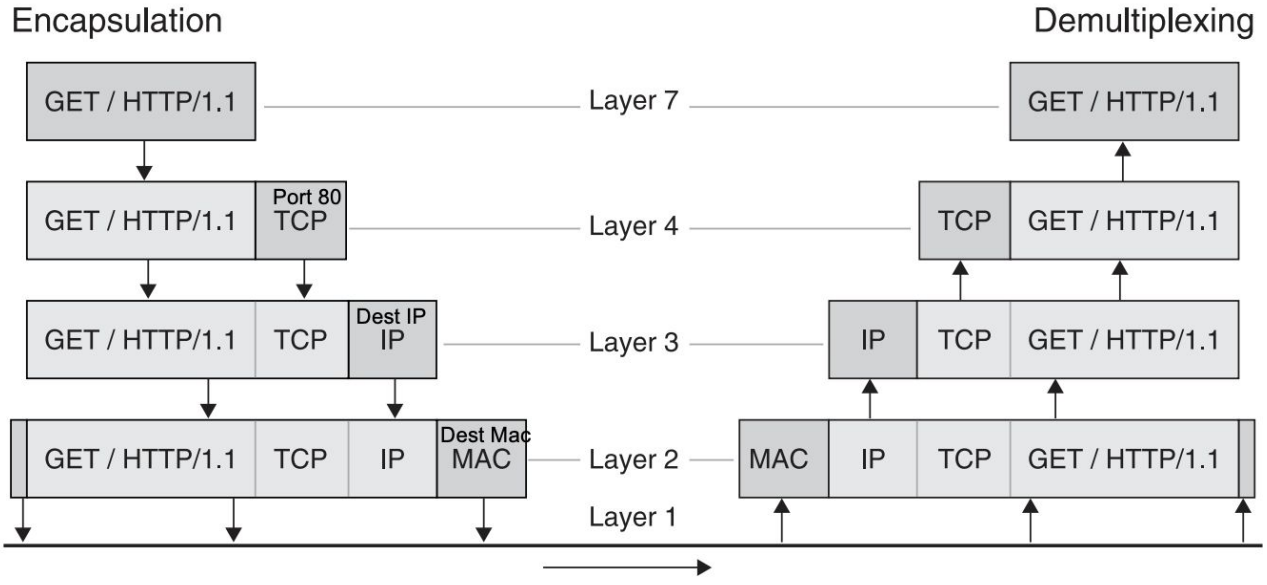
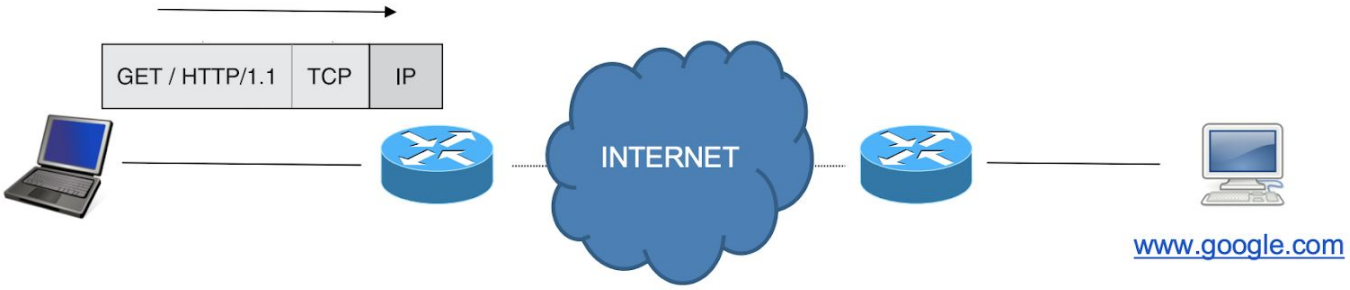


Figure 2–2. An HTTP "GET" request, shown in the framework of the OSI model.

By default, an HTTP query will be encapsulated with a TCP packet. By default every router listens to port 80, then you add the destination IP and the destination MAC address. When a person receives it, it can do the inverse steps.

In reality, most of the time we are not querying our LAN, so we would normally send our IP packet to our router and he would distribute that.



Traditionally, if we are not using Tor browsing, the header will be in the clear, thus the source and destination IP addresses.

Sniffing

Sniffing: listening to the network conversations which are not intended for you.

- Sniffing can also be used for a benign purpose, debugging or traffic diagnostics (intrusion detection systems).

Physical Layer (layer 1)

When sniffing layer 1 we might sniff:

- The wire
- Air
- Router
- Switches

We can decide how we want to get the information depending on:

- We want a passive or active attack
- We are in the network or not

From inside the Network

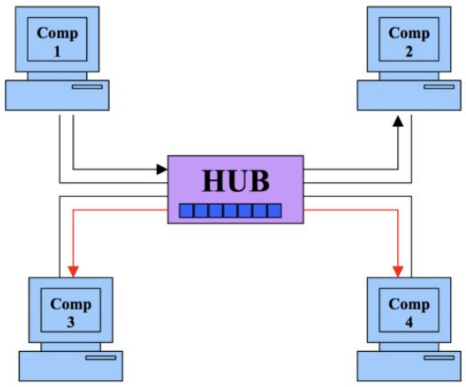
Passive Interception (in the network)

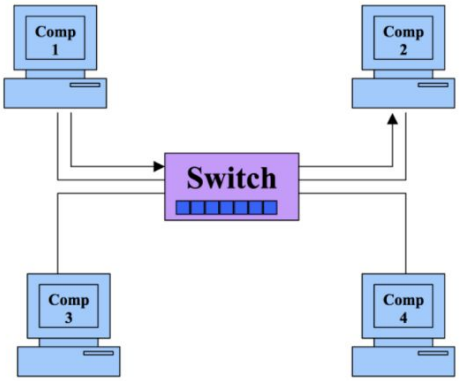
Putting our network card in a promiscuous mode so that we get all the packets of the network rather than just the ones destined to us (which is what the network card does, filter out packets in the network and only pass to the operating system the ones that are intended for us).

This works for networks using **Hubs and Wi-Fi** networks but **not for** wired networks using **Switches**. Switches are like a simple gateway between traffic and a specific MAC address that we have specified for that specific port, th at means that although we put our card in promiscuous mode, we only receive data intended for us.

Hubs vs Switches

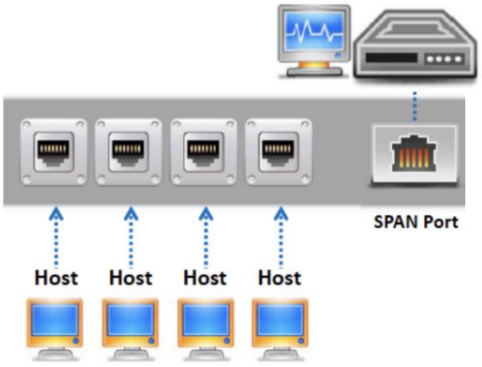
HUBs broadcast the same data out each of its ports and let the devices decide what data they need.
Switches are more advanced and forward the data only to the devices that need to receive it (to optimise the network).





With HUBs everything sent from Computer 1 to Computer 2 will be received by computer 3 and 4, with a Switch it will not.

Active Interception (what we can do if connected to a switch)



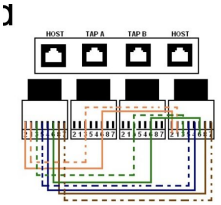
If we have access to the switch we can physically connect to the SPAN port in the switch, which mirrors all traffic through the switch. It is normally used for:

- Network diagnostics
- Intrusion Detector

Plugging into the SPAN port can let an attacker see all network traffic.

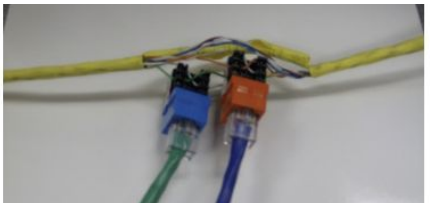
From outside the Network

Wired Networks



In line network taps (layer 1): devices, cause a brief disruption as cable needs to be disconnected and reconnected.

Vampire taps (layer 1): Pierce the shielding of copper wires to provide access to the signal within. Although, we may bring down the link, it is typically used by Telecommunication Engineers.



Wireless Networks

To tap wireless networks we can:
Warwalking: tools like netstumbler to walk about and when you are within reach connect to the network (scan the place for network and connect).
Wardriving: same as walking but driving around.
Warflying: with drones.
Rogue Access Point: a wireless access point that has been installed on a network without explicit authorization from a local network administrator. This means that all the traffic that is supposed to go to the correct router first goes through your computer.

- Hardware: a router
- Software: a laptop with virtual network.

They are usually open and try to mimic a well known WIFI Network ID.

Data Acquisition (layer 2)

Given that we have chunks of bytes from layer 1 we must now interpret the meaning of these.

We find that these have information about layers above them like ARP, MAC, IP...

pcap: basis of wireshark & tcdump

Pcap is a library that helps you process all the communication interfered. They are the basis of tcpdump and wireshark.

1. Capture only packets of the icmp protocol

```
tcpdump -i eth0 icmp
```

2. Capture ssh packets between two hosts

```
tcpdump src <x> and dst <y> and port 22
```

Each time data has to be fetched, user code has to trap to the OS kernel and have buffer filled. The solution offered by the **Berkeley Packet Filter** is filtering commands written as commands for a virtual machine interpreter run by the kernel. Just-in-time compilation of filtering regular expressions is used to decrease overheads

We have assumed that the bytes that we obtained are from a wired network, but what if they are from a wireless network. In a wireless network:

- If the network is open, we can do the same as with a wired connection
- If the network is encrypted:
    - Using WEP (at data layer): it can be cracked
    - WPA-PSK or WPA2-PSK: still crackable if you are in physical proximity and easy if you are inside the network.
    - WPA-enterprise or WP2-enterprise: it is difficult but not impossible.

  Once we decrypt the data we can use libcap.

Traffic Analysis

Protocol Analysis

Analysing the traffic sent to identify the protocol being used. Given that the most common protocols are public, therefore we can identify what protocols two agents are using and give meaning to the packets that are being sent back and forward. In other words understand the semantics of the information being transmitted to be able to interpret it. For instance if they are sending IP, TCP, ICMP and thus infer what is going on.

Packet Analysis

We try to disclose the information contained by inspecting traffic. To do so we can do:

- **Pattern Matching:** Identify packets of interest by matching specific values within the packet capture. *Ie we want to find packets that come from a particular source.*
- **Parsing Protocol Fields:** extract the contents of particular protocol fields.
- **Packet Filtering:** separate packets based on the values of fields in protocol metadata.

Flow Analysis

Flow analysis is the practice of examining related groups of packets in order to:

1. **Identify patterns** *E.g. repeated communications. E.g. for all computer in this room, what are the destinations, is there a pattern.*
2. **Isolate suspicious activity and discard irrelevant data** *E.g. you want to isolate some traffic sent to some IPs that we believe are malicious.*
3. **Analyse higher-layer protocols** *e.g. reconstructing segmented TCP packets and get the full picture of the protocol encapsulated in it, e.g., HTTP.*
4. **Extract data** *e.g., a binary file to be analysed. We look at layer 7 by taking information from the rest.*

Flow analysis tools **reconstructs the full duplex contents of that stream** from beginning to end.
Duplex: not only what you sent to the server but what the server replied back

Flow Analysis Techniques

By doing flow analysis we can:

- **List conversations and flows:** list all conversations and/or flows within a packet capture, or only specific flows based on their characteristics.
- **Export a flow:** isolate a flow, or multiple flows, and store the flow(s) of interest to disk for further analysis.
- **File and data carving:** extract files or other data of interest from the reassembled flow.

L3 Spoofing and Jamming

Jamming at the link layer

The link layer is broadcast-based. Users must be polite and must talk one user at a time.

When we want to **jam** this layer we hog the broadcast medium so nobody can talk. This can be done on ethernet and wifi. *Talking very loudly between people.*

Portable jammer

Wifi
3G/4G
5-20 meters

£100-200

Jamming the Link Layer is easy, for example Ethernet.
Throughput: a performance measure of how many frames of data are successfully delivered per unit of time.
Contention: sharing the broadcast medium across a number of devices. (Broadcasting from multiple devices)

What we can do is to try to transmit information and then stop transmitting. Then I take random time to try again, the more people we have doing that, the less throughout we have.

Throughput of randomized multiple-access drops drastically as **contention** increases.

Jamming at Network Layer = DoS

To do a Denial of Service attack (DoS) we want to send so many packets to overwhelm a server or network link, this can have a huge impact. For example:

- AWS makes an estimated $29--31K every minute

- Paypal was attacked by Anonymous & lost nearly £3.5 million

The are many forms of DoS, several include spoofing.

Spoofing

Spoofing: pretending to be somebody that you're not.
But what is *somebody* on the internet?

- An IP address (network layer)

What is *somebody* on the LAN?

- A Mac address (data link layer)

Each layer has its own address:
IP address at network layer (logical address)

- Used by applications and the socket interface

Socket interface: an abstraction that help programs to develop applications that do networking. It normally creates TCP (stateful) / UPD (stateless) packets for you allowing you to simply send data through the socket without having to worry about the rest.

MAC address at link layer (physical address)

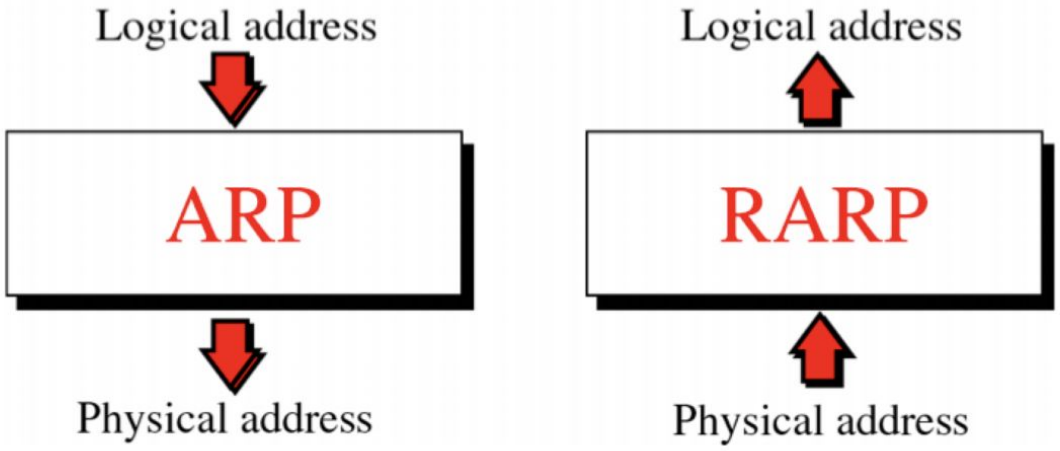
- Ethernet header contains the MAC address of the source and the destination computer. Used for link local communications, i.e., on each IP hop.

ARP

ARP (Access Resolution Protocol): is a resolution protocol that takes in a logical address (IP) and returns the physical address (MAC).

The local area network variant of DNS kindoff.

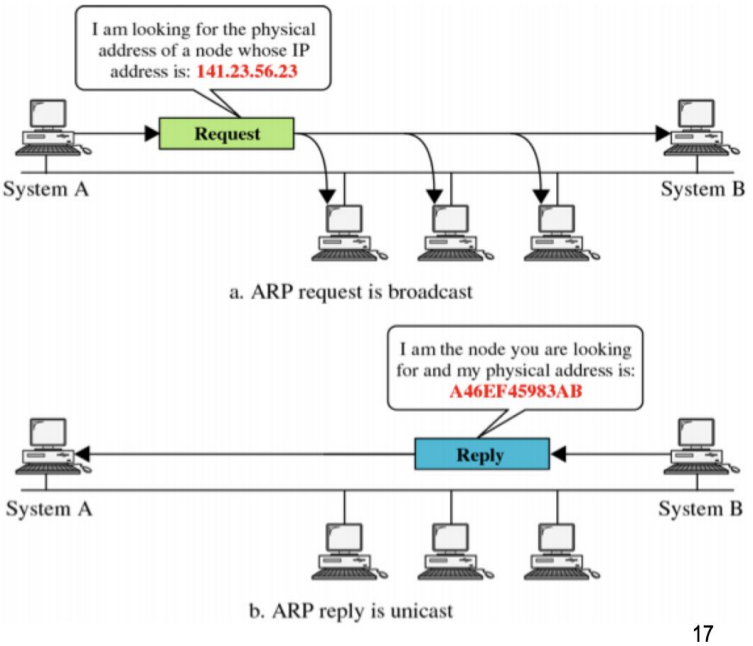
> At the link layer, IP is encapsulated in an ethernet frame.
>
> But what ethernet destination address to use for a given IP address?



Physical Address = MAC Address, Logical Address = IP

How ARP works

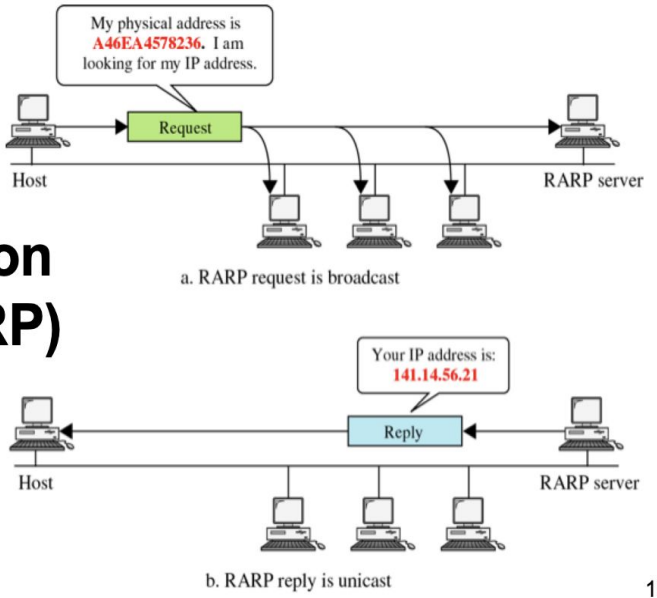
ARP operation



17

ARP request: a broadcast message to the whole local network which asks who has that specific IP address and to please return their physical address. Now that you have the machine's you want to talk to MAC address you can now have a unicast conversation (non-broadcast) with that specific host.

RARP operation (reverse of ARP)

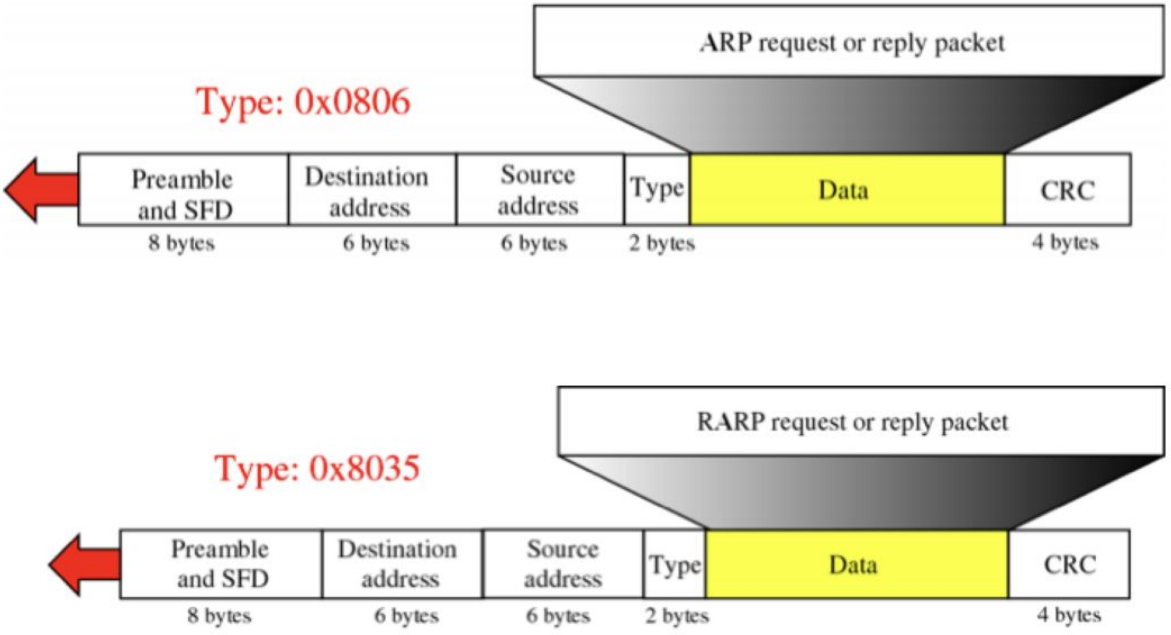


18

Reverse ARP: I know what my physical address is, I want to know my IP.

This protocol is now obsolete,Nowadays you either put the IP into the device manually (statically) or we use a protocol called DHCP.

ARP and RARP are ethernet msgs



SFD: start frame delimiter, says that here starts the ethernet msg.
Type: ARP or RARP
CRC: circular check

In the response in the data we will have an IP packet.

Algorithm:

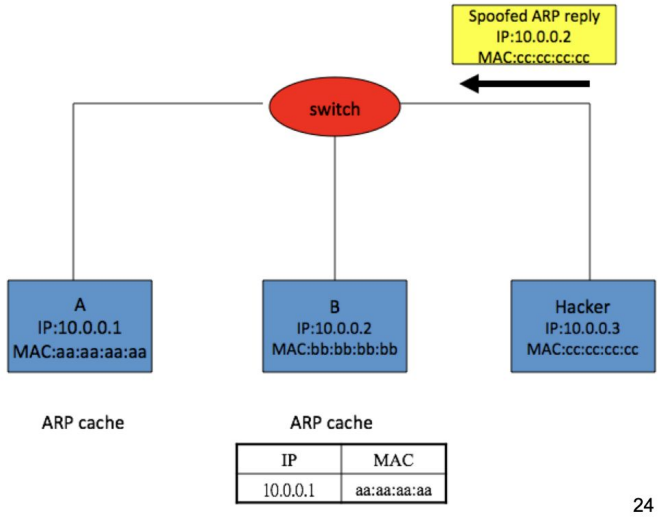
1. Get IP address of target.
2. Create a request ARP message
    a. Fill sender physical address
    b. Fill sender IP address
    c. Fill target IP address
    d. Target physical address is filled with 0
3. The message is passed to data link layer where it is encapsulated in a frame.
    a. Source address: physical address of the sender.
    b. Destination address: broadcast address.
4. Every host or router on the LAN receives the frame.
    a. All stations pass it to ARP.
    b. All machines except the one targeted drop the packet.
5. Target machine replies with ARP message that contains its physical address in a unicast way to the original sender of the ARP broadcast request.
6. Sender receives the reply message and knows physical address of the target.

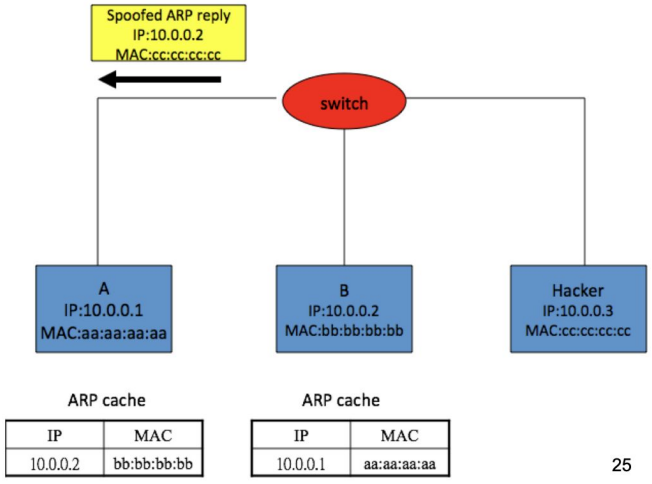
ARP Cache: To avoid having to send an ARP request packet each time, a host can cache the IP and the corresponding host addresses in its ARP table (ARP cache). Given that ARP is stateless, whenever ARP receives an ARP request it will update its Cache even if it didn't send out a request.

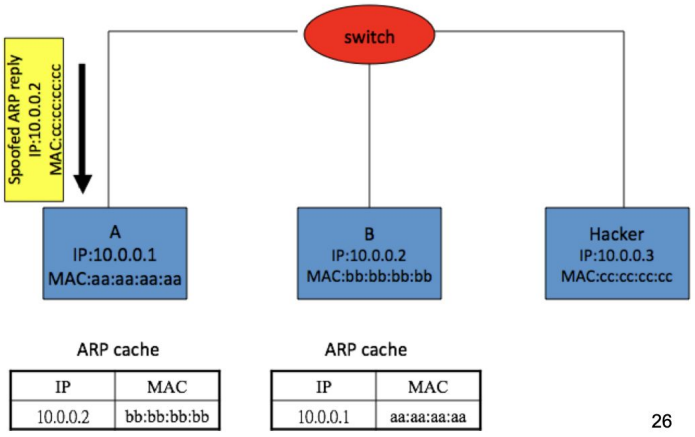
Each entry in the ARP table is usually “aged” and contents are erased if no activity occurs within a period.

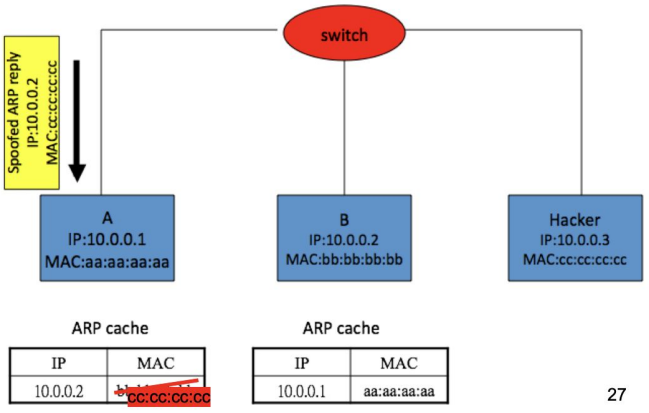
ARP Spoofing Attack

ARP Poisoning: to attack the system we can create **spoof ARP replies**. For example make a computer send packets that are originally for A go to B. A will have no idea that the redirection took place.









Then all packets intended for B will go to the hacker.

Complications:

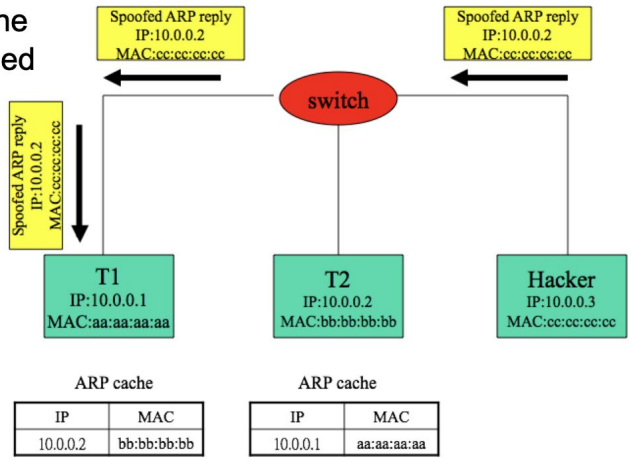
- Cache entries expire so there is a time-window in which we can be malicious or continuously spoof.
- Some systems may try to send unicast ARP to confirm or update cache

In both cases, we can re-send ARP spoof packet (once every ~40s is sufficient usually) and the attack is still valid.

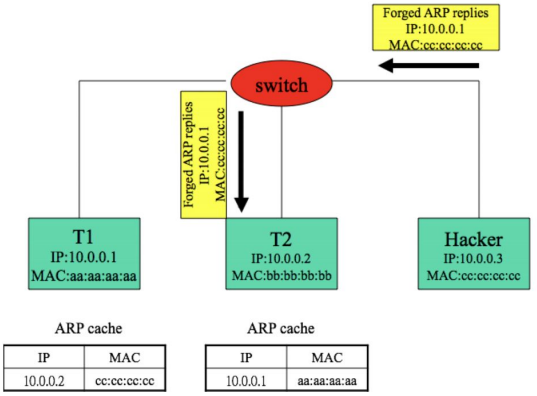
Man-in-the-Middle attack with ARP spoofing

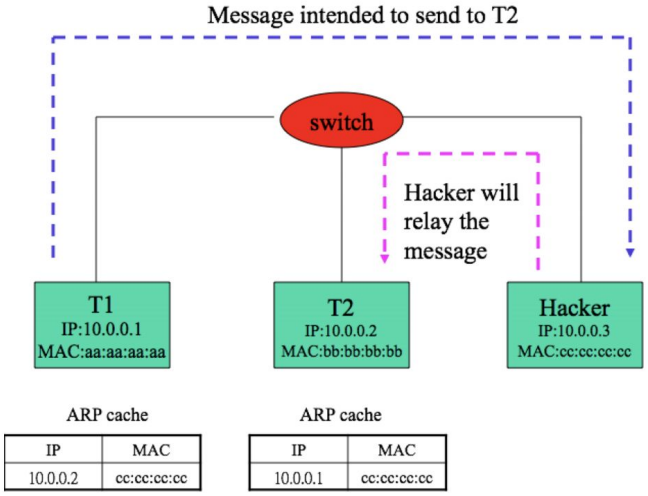
Goal: **insert** hacker’s computer **H** in between conversation of **A** and **B**. **A** and **B** should continue conversing but hacker has complete access to all their packets. If we are able to put ourselves between two computers, we can sniff what is sent. If we manage to do ARP spoofing we manage to get the traffic of the network.

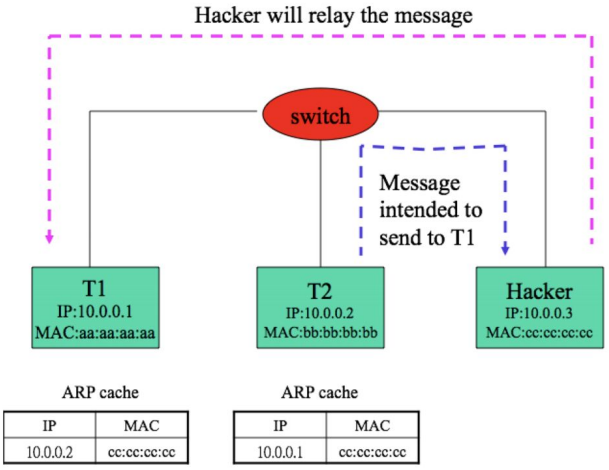
T1's cache is poisoned



T2's cache is poisoned







MAC Flooding with ARP Spoofing

Switches have an internal table which maps switch ports to MAC addresses.

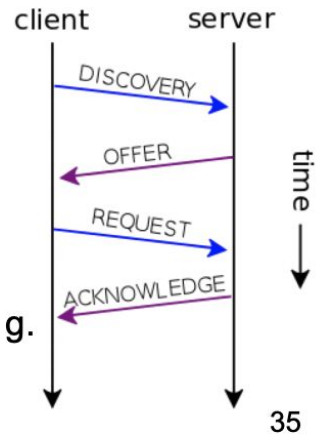
Switches have a finite memory. In a MAC flooding attack, a switch is fed many Ethernet frames, each containing different **spoofed source MAC** addresses to consume the limited memory in the switch and force significant quantities of incoming frames to be flooded out on all ports. Thus it starts behaving like a hub.

```
2960-1#show mac address-table
Mac Address Table
-------------------------------

Vlan Mac Address    Type    Ports
---- ----------- -------- -----
1    001d.70ab.5d60 DYNAMIC  2
1    001e.f724.a160 DYNAMIC  3
```

Active Interception for Sniffing!

DHCP Starvation and Rogue Server



Dynamic Host Configuration Protocol (DHCP) is a protocol that automatically provides clients with IP addresses and other related config (e.g. default gateway)

What the DHCP protocol does:

1. Discovery - Broadcast message to discover the server. *Who is the DHCP server in the network?*
2. Offer - The server sends a unicast message of offering an IP address to the client.
3. Request – Client accepting the IP offered
4. Acknowledge – Server acks. the IP and sends other related configuration information like the IP address of the gateway, network mask, IP address of DNS server and so on.

This can be used to attack.

DHCP Starvation

In a **DHCP starvation attack**, an attacker broadcasts a large number of DHCP_REQUEST messages with spoofed source MAC addresses. Then the DHCP server starts assigning IP addresses to all those IP addressed but at some point the DHCP server runs out of IP addresses and legitimate clients cannot get an IP address.

DHCP Rogue Server

Once the available number of IP Addresses in the DHCP server is depleted, network attackers could set up a rogue DHCP server and respond to new DHCP requests from network DHCP clients. This can be done by running a program that whenever senses a DHCP requests, sends a response.

DHCP spoofing attack
The Rogue server starts distributing IP addresses and other TCP/IP configuration settings including Default Gateway and DNS Server IP addresses, which can now point to an IP address controlled by the attacker. Facilitates MitM and Sniffing attacks.

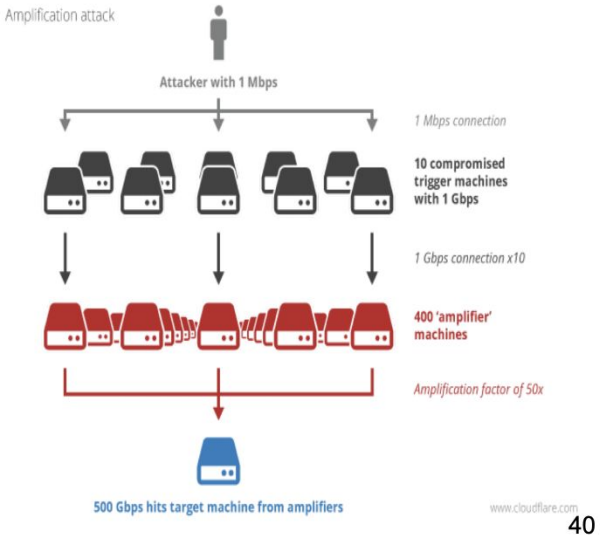
Spoofing attack 2: Smurfing

A **smurf attack** is one, where a computer S can get all other computers in the network to send an IP packet to an intended victim computer V. S can do this by sending an ICMP Ping request to the broadcast address of the network. All computers think it is destined for them because it uses the broadcast address. Ping protocol requires a response to be sent back. If S spoofs V ’s address in the source field, these Ping responses get sent to V – a smurf attack.

If we broadcast a ping to our LAN and spoof the source address (ie set source the person you want to attack and destination broadcast), then all the hosts in the network will reply to the spoofed address which will create a Denial of Service for our source address.

Example of **amplification attack:** small amounts of traffic generate large amounts of traffic in another direction.

Small amounts of traffic are converted into large amounts of traffic. This is known as **amplification**



40

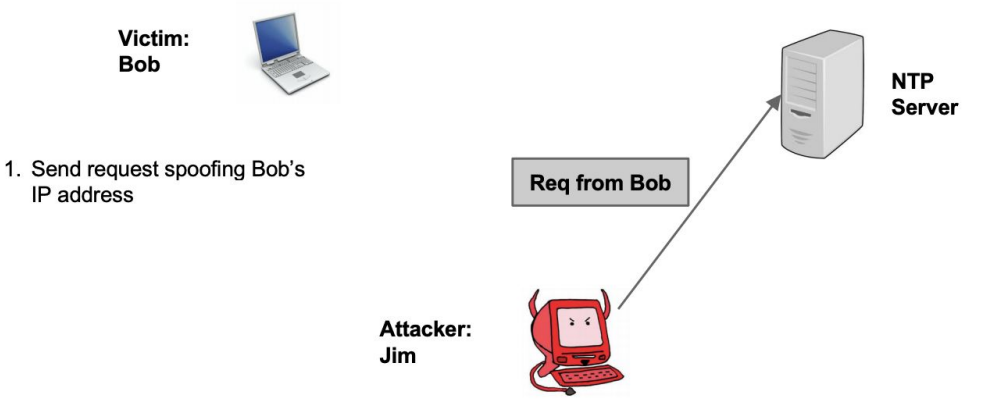
Spoofing attack 3: NTP DoS Amplification Attack

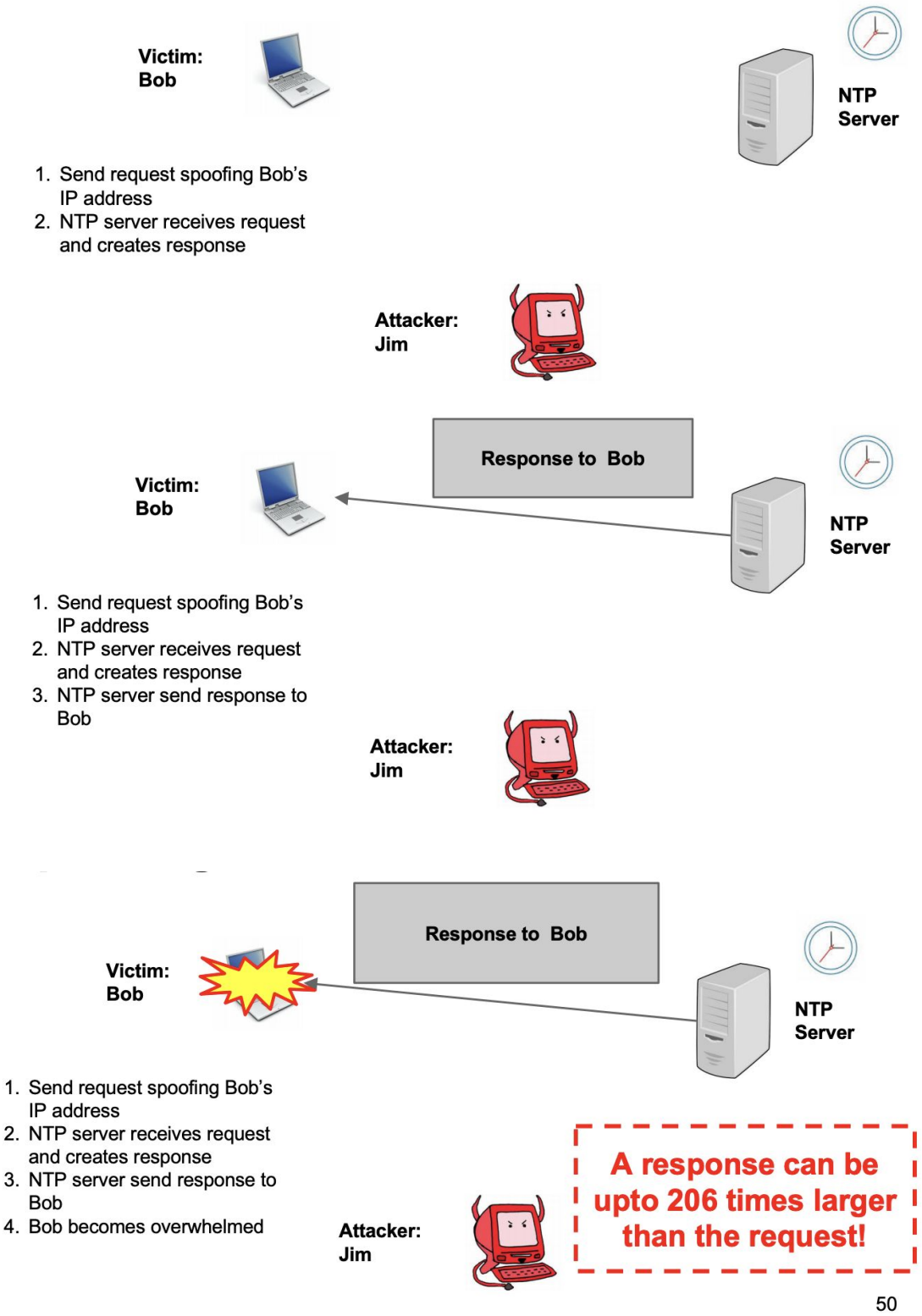
Network Time Protocol (NTP): allows computers to synchronise their clocks. It works based on UDP.

UDP: you send a packet and a port and behind that port there will be a service listening to those packages, but no connection (*sending a letter).*
TCP: establishes a connection and then use that connection to talk *(opening a whatsapp message with someone online)*.

NTP request are very small but the response can be very big. If we spoof the IP address to send response to victim.
The **$monlist** command returns last 600 hosts. Versions of ntpd prior to 4.2.7 are vulnerable.

In this way, by sending a small request we can attack a machine by having it receive a very large response.





To make this attack even more powerful we can query not one but **multiple** servers.

- In Feb 2014, CloudFlare witnessed a 400 Gbps NTP attack Used 4529 NTP servers

We can do something similar if we can use a botnet.

Botnets

Botnet: networks of infected computers that you can control (install some malware) and then you have them do whatever you want. The malware establishes a connection with a command and control server which asks

what it has to do next. You normally use HTTP or HTTPS so that it is hard to know what is going on. Botnets are used for distributed denial of service attacks and other types of attacks (spam, data theft, etc) by making them work at the same time targeting a specific machine.

Formally:
A botnet is a logical collection of Internet-connected devices (computers, smartphones or IoT devices) that have been infected and are controlled by a third party. A controller software (known as “command & control”) is able to direct the activities of each compromised device (known as a "bot”) using network protocols like HTTP. Botnets are used for distributed denial of service attacks and other types of attacks (spam, data theft, etc)

E.g. Miriai scandal, CCTV cameras were infected and used as a DDoS.

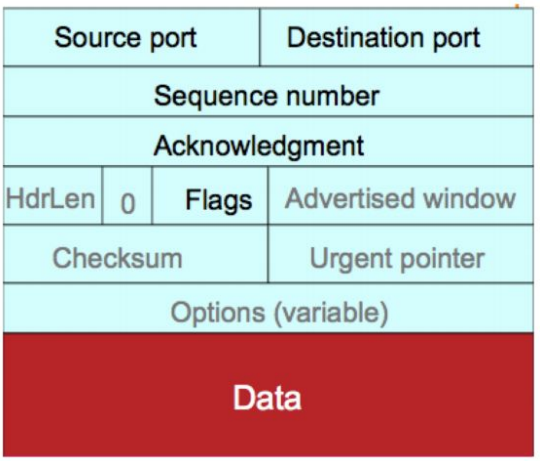
Spoofing Attack 4: SYN flooding

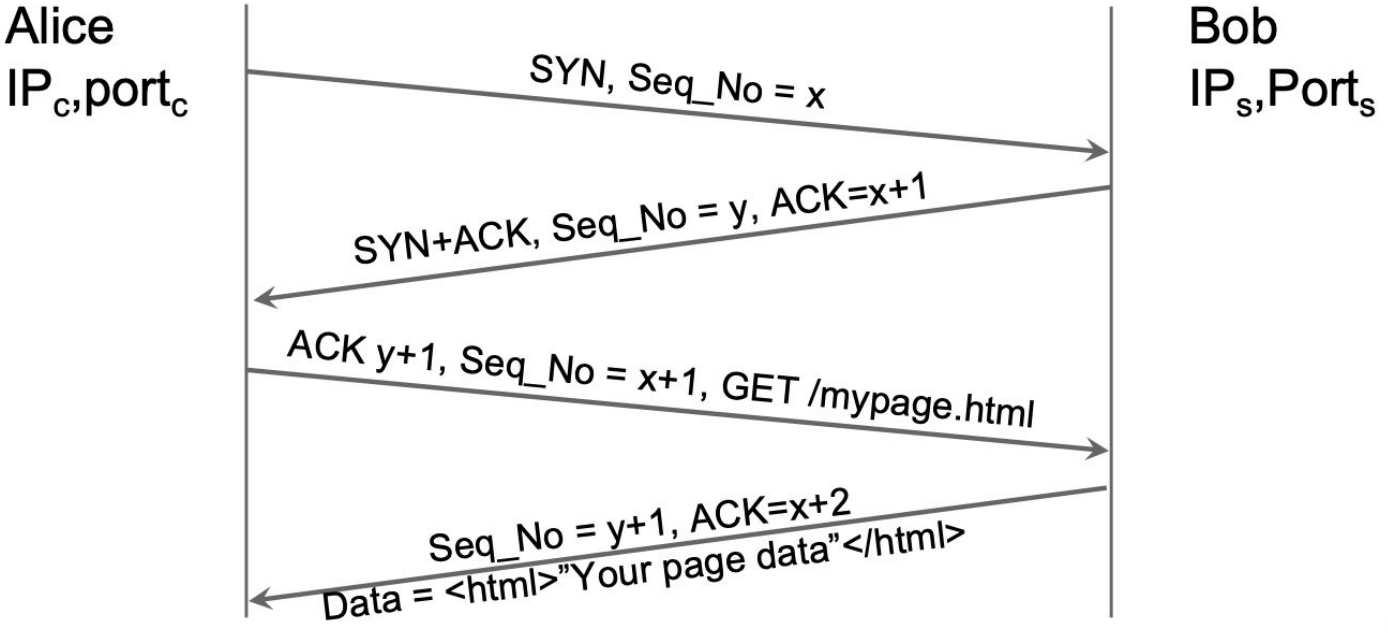
TCP (Transmission Control Protocol) defines a connection of ordered *sequence of bytes, over an unordered IP network.*

TCP works at the transport layer over IP

IP address + port number = connection

In the connection setup:





1. SYN message stating that you want to establish connection (with SYN flag) and Sequence_Number = X.
2. Response from bob with ACK message saying x + 1 (I know your seq_no and acknowledge that), Bob’s seq_no y and SYN + ACK flags.
3. Response from Alice, Acknowledging your acknowledgement by updating your sequence_number, updating her sequence number and sending a query.

From this number the connection is established as they know each other's sequence number and Alice and Bob can send packages to each other in an orderly fashion.

4. Response to the query with the payload, updating my sequence number and acknowledging your precious packet.

SYN flooding: works by not responding to the server with the expected ACK number (3). The malicious client can either not send the expected ACK, or more effectively ***spoof the source IP address*** in the SYN, causing the server to send the SYN-ACK to a falsified IP address - which will not send back and ACK because it "knows" it never sent a SYN.

Even though you didn't send the response, the server remembers the connection, therefore the server will wait for the missing ACK for some time, however, the resources bound on the server may eventually exceed resources available and the server cannot connect to any clients.

Lecture 4 Identity Attacks on the Internet

Spoofing: pretending to be somebody else (having an address that is not meant to be yours).
Hijacking: taking something that belongs to somebody else (especially at runtime).
Poisoning: contaminating some source of information (that is usually trusted to be good).

These three types of attacks compromise integrity.

Hijacking

We can use hijacking to hijack connections to:

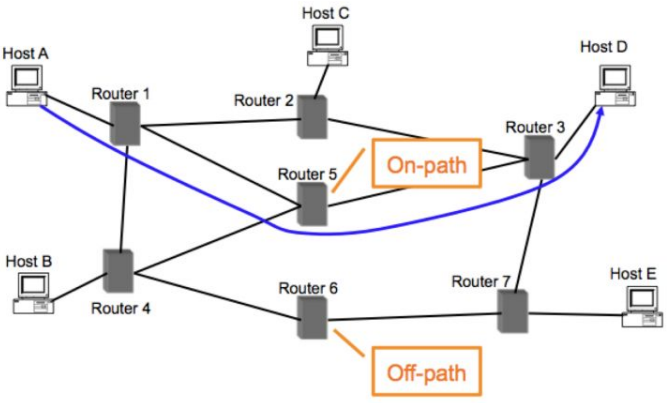
- Disrupt, or jam communications
- To insert false or malicious data
- Similar to spoofing except that, in general, hijacking takes over an **existing** connection, spoofing starts a new unwanted connection.

On-path adversaries vs Off-path

On-path means that you are in the *path of the packets* from one host to another (in the example Router 1, 5 and 4). If you are on path it is relatively to take over the connection as you can see the connection between two hosts, you can see the TCP Sequence number, the ports they are using and use this information to try to participate in this communication.

- On-path adversaries are more powerful.
- Inserting false data is very difficult if off-path.
- On-path adversary can use connection state (difficult to do at scale because of thousands/million of connections going through a router).

Note the terminology is from the point of view of the hosts we want to target. To compromise the connection between hosts A and D we must be on-path of their connection.



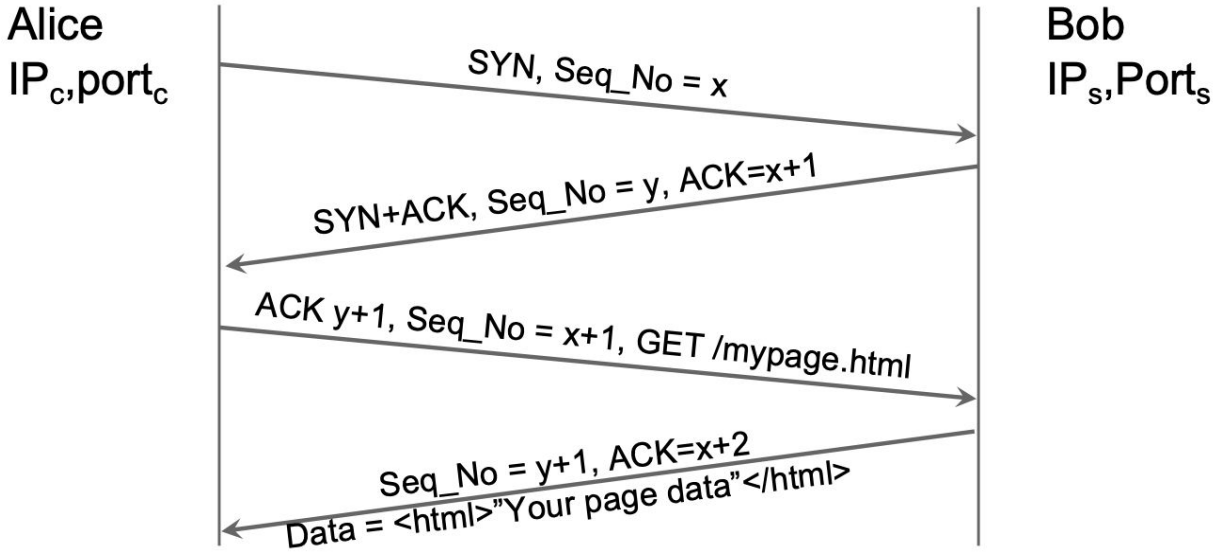
The takeaway is that being on-path or off-path deals with the difficulty to hijack the network.

When you want to send a packet to say Host D, you as Host A send you packet to Router 1. Then Router 1 decides where it wants to re-route that package given the shortest path it knows to the host D.

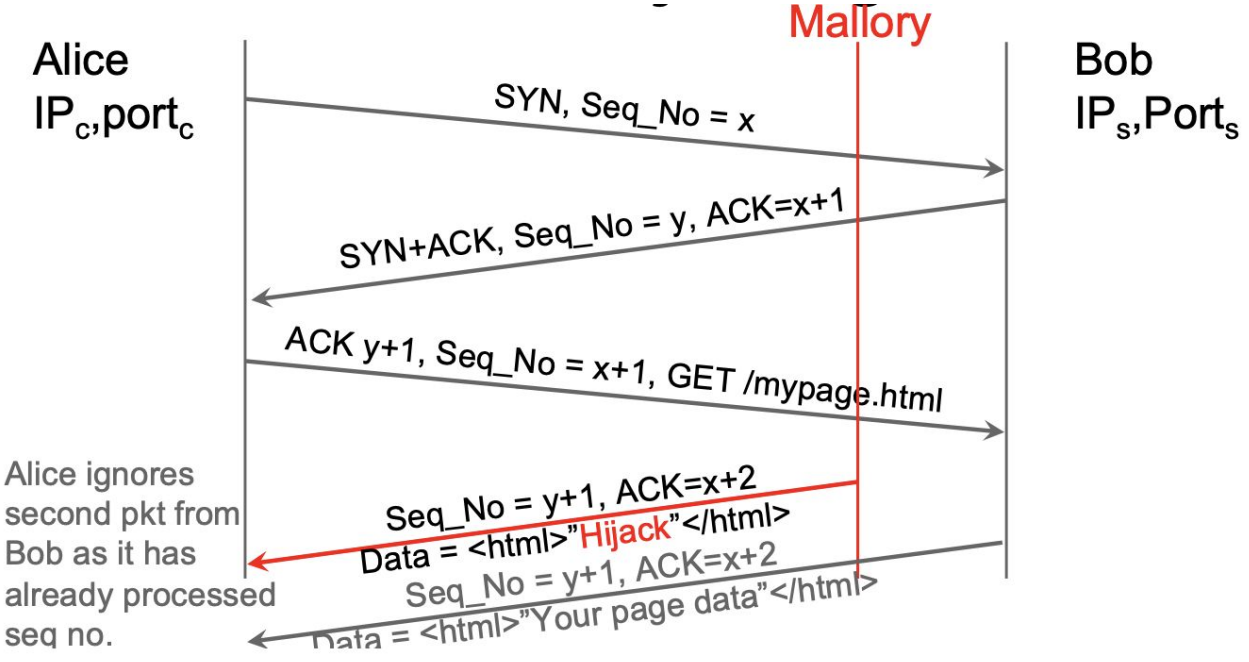
TCP: Transmission Control Protocol

TCP Connection: defines a connection of an ordered sequence of bytes over an unordered IP network.

- IP address + port number = connection
- The sequence number defines the order of packet payload within the larger stream/connection.



To hijack a connection, you want to know the initial sequence number as you want to make sure that when you are sending messages in that connection, you are sending the correct number and the correct sequence. You must send packets "which make sense" to the receiver.



The initial sequence number for Alice is **x** whilst the initial sequence number for Bob is **y**. In this was we can keep track of the packets sent by both connections.

Ack number the sequence number he expects (or that is yet to be sent) in the next round (note, not the same as the flag)

After the three way handshake there are no more processes in the handshake. Nevertheless, the ACK still tells us up to where he is up to date with information. Furthermore, if Mallory intercepts the response correctly, Alice will then acknowledge that to Bob and Bob will think his packet arrived correctly. In this way, the connection is still alive and both parties may not know about Mallory.

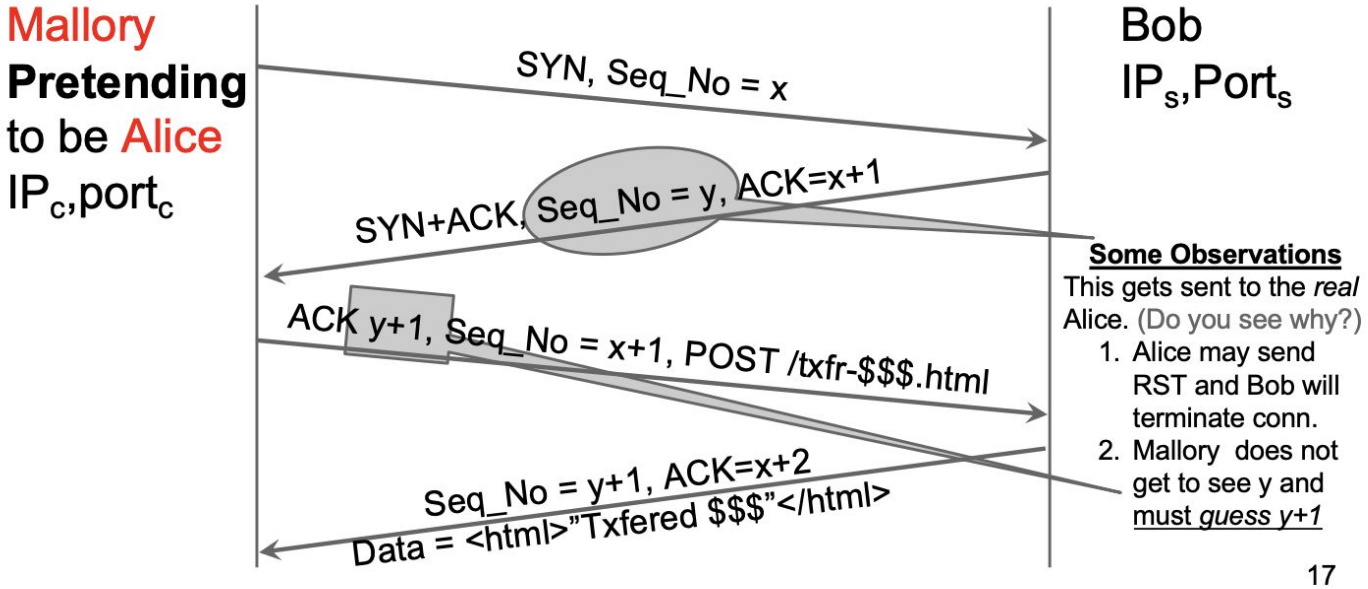
Mallory can hijack the stream by:

- **To spoof client:** authentication may happen at beginning of connection. By hijacking connection after authentication, Mallory can leave a record of bytes as an “Authenticated” user:
    - e.g. HTTP POST /transfer-money £100,000
- **To spoof server:** Insert false data from server to client (attack as shown in timing diagram above)

What can Mallory do off-path?

In reality we don't know even if there is a connection. We can try to spoof the connection, sending a message to Bob pretending to be Alice. For that to work Mallory’s packets must fit the right sequence number Bob is expecting from Alice, if not he will ignore the packet.

Initial Sequence Number Attack



Mallory cannot see the second message as the Seq_No = y is sent to **Alice** since Mallory is spoofing **Alice**. Mallory has to be very quick in guessing y+1. She has to be quick because if Alice gets the TCP message and replies on time, she will send a RST (reset) flag message because she did not establish that connection and Mallory’s next packet will be discarded since the packet with the correct sequence number has been delivered.

Guessing initial sequence number

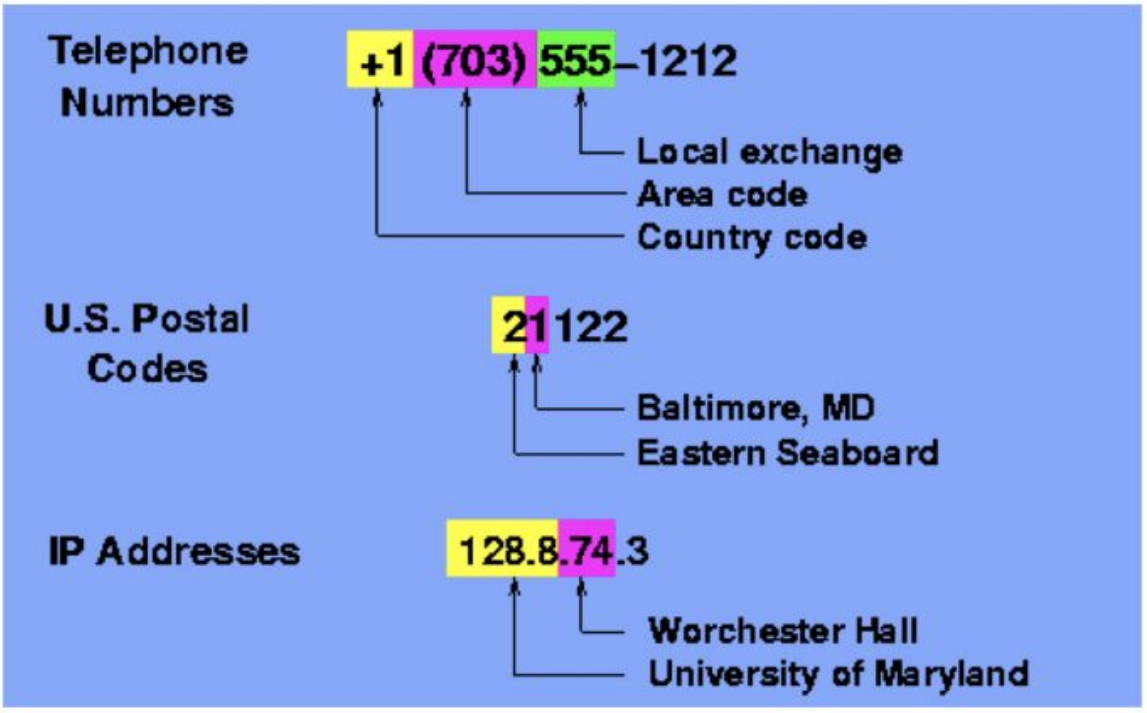
Since we want all computers to be able to communicate with each other we must use an international standard. The **sequence number size is limited to 32 bites (4 bytes)**, there are lots of numbers but that is the maximum. Since it is a finite number it will restart, these repeat occurs every 4.55 hours.

The TCP standard sets a way to establish how to create a TCP sequence number. This may deviate from OS to OS. It might take time but there is a way to guess an initial sequence number.

Note even if you send a packet with a sequence number in the current seq number plus a window (in the future), you might get lucky and your packet might be held until its the correct sequence number.

BGP Route Hijacking

We can look at routes as cars travelling as packets from X to Y, from a high level perspective of all packets moving around it can be quite complicated. Although there might be some guidance (signs) these might be static. Normally what we do is we look at different granularity levels. We do this with the IP address, some part of the api address can give us different information about where we are trying to go.



We want to learn which is the best route. We use a protocol used BGP to do that. We can hijack (change) either the **route** or the **destination** of a packet.

Simplified intro to BGP-1

Border Gateway Protocol controls the routes packets take through Autonomous Systems (ASes).

Autonomous System /inter-domain routing: a number of routers under the same domain/administrated by the same entity.

- E.g. Router

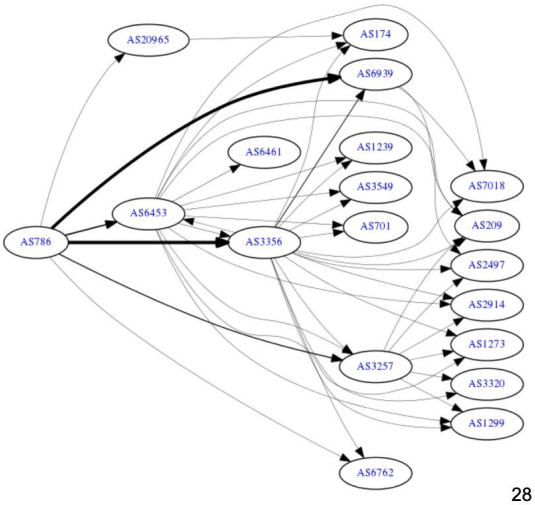
Autonomous systems advertise the prefixes they can serve

E.g. 137.73.0.0/16

Prefix

16 bits

/16 remember - tells us how long the prefix is (in this case 137.73 since it's /16. if it were /32 it would be only one IP address



28

What it is saying is that I can talk to all IPs which are within these, ie all of them start start 137.73.X.X

- The ASes go to the longest prefix matching to select which neighbours to route through (ie the one that has the most accurate, longer as this means it is closer to the address we want to arrive to).

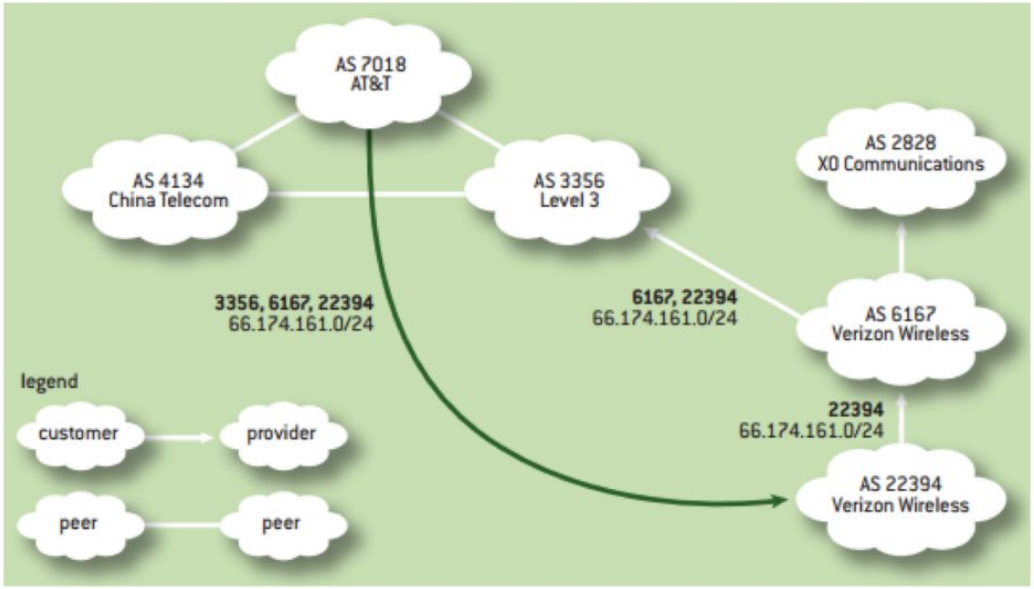
ASes forward their advertisements so you may end up knowing that through certain hops you can get an AS that can connect you to the domain you want to arrive to.

Routers will build tables of the prefixes they can serve as well as tables of the prefixes nearby routes can serve to know if they should handle a response or who they should forward it to.

- AS7018 advertises 66.174.161.0/24 and it gives the sequence of other AS that lead to the advertised network (3356, 6167, 22394)

- AS 22394 is the "origin"

- Note the green line represents AS7018 advertising a network originally advertised by AS22349 but they are not directly connected!



As we can see AS7018 advertises an IP that he does not possess nor is directly connected to, but as it knows a route in which it can serve that, it can advertise it.

What if AS lies?

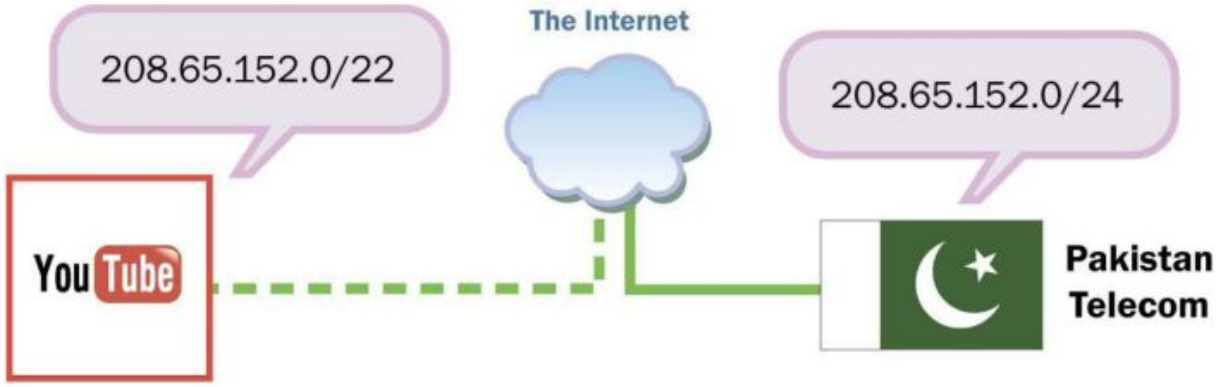
We can hijack the prefix of our addressed. Given that BGP prefers the longest prefix match, say UCL advertises 137.73.0.0/24 instead of 137.73.0.0/16 then all packets intended for KCL would go to UCL (given that 137.73.X.X is kings).

Note that there is a tradeoff when advertising longer prefixes we are discarding other possible IPs since we are making more bits static from the IP. Meaning that we will not cover as many addresses.

Routing Black Holes

We could advertise as if we could serve some services so that their traffic comes to us. If we cannot serve this packages, then we create a black hole in which people are waiting for you to provide a service you advertised which you cannot provide.

In some cases this can be used for censorship. For instance a Pakistani Telecom wanted to censor YouTube and advertised that it could serve YouTube server response so that all the YouTube requests went to it. Unfortunately, the admin published that it could do this to everyone, meaning that all the traffic to YouTube severs was directing to the Pakistani Telecom thus creating a DDoS attack on themselves.

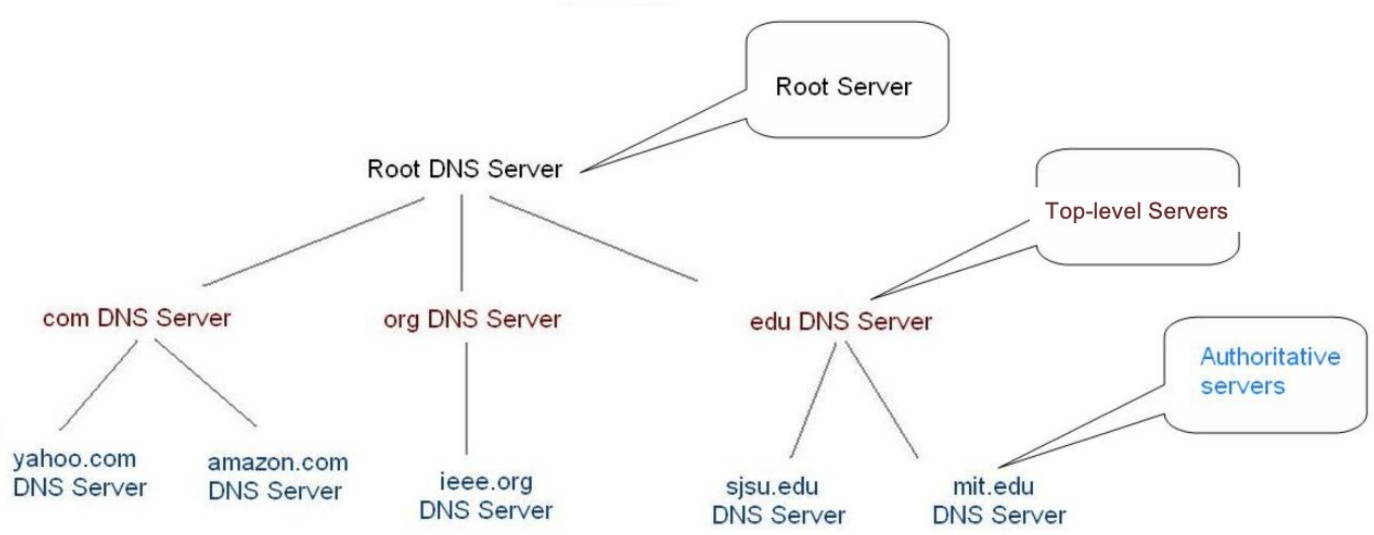


DNS Cache Poisoning

It is easier for humans to remember a URL than an IP address, also it is easier that all the servers are within the same URL and not under the same IP as then we can do load balance. Essentially DNS maps www.kcl.ac.uk to 137.73.118.10.

DNS Hierarchical Structure

DNS has a hierarchical structure. It first looks at the end of the domain (.com, .org, .edu etc)



A DNS Resolver performs lookups to map names to IP addresses (POSIX Command: **$ gethostbyname**) or IP address to names (POSSIX Command: **$ gethostbyadd**), it does this by contacting nameserver(s). This is very efficient nowadays.

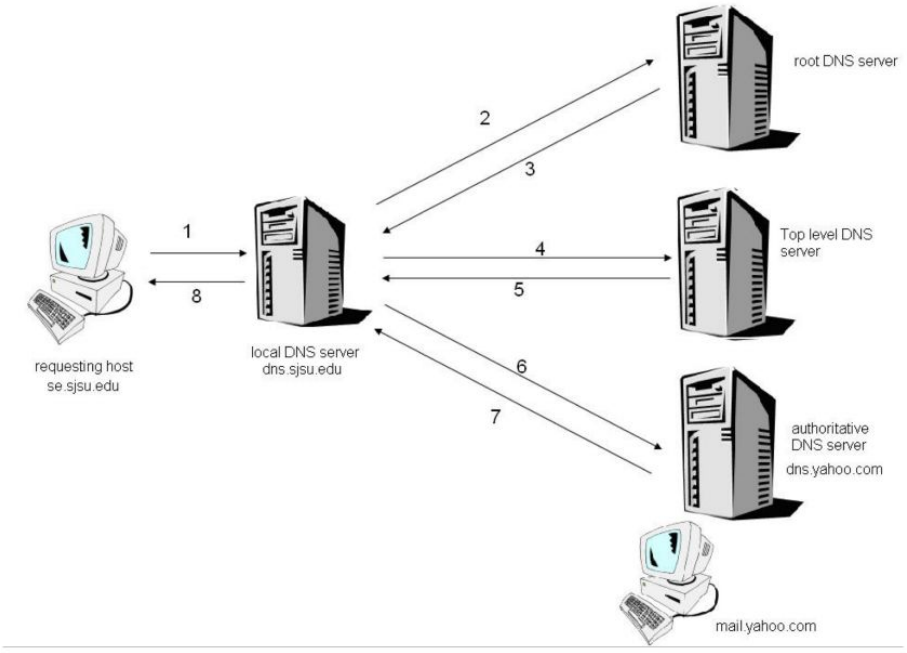
DNS Name Resolution

There are two ways:

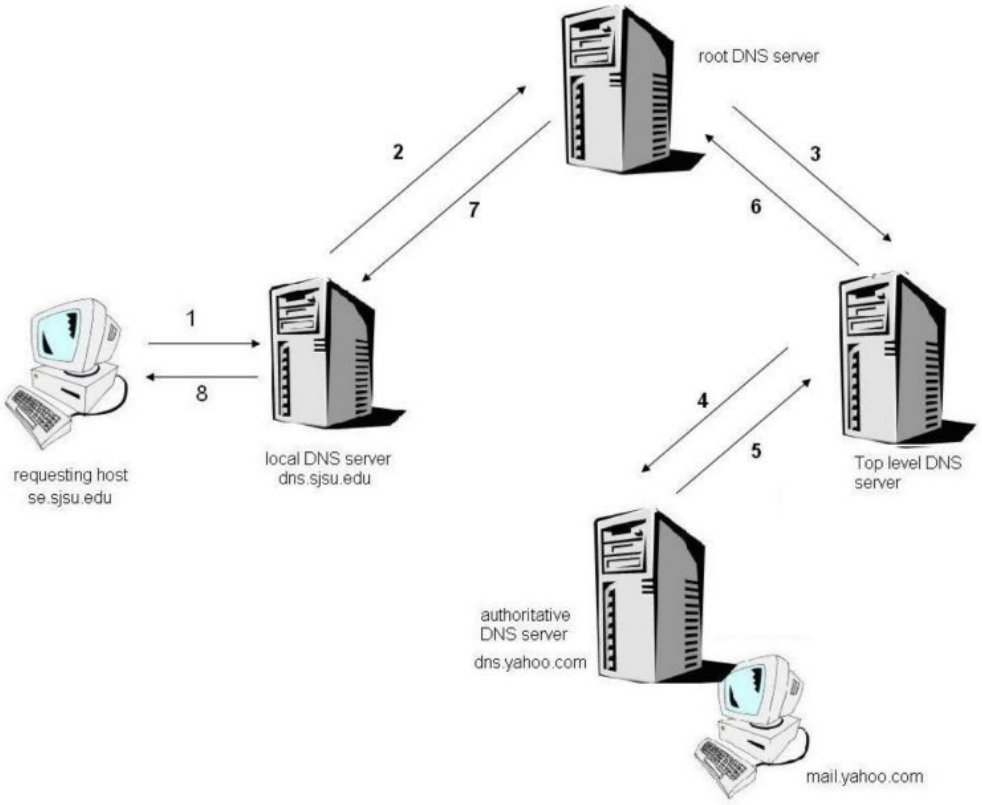
- Iterative
- Recursive

Iterative

You ask your local server and then ask the root server, which will give you a response, then you will ask to that server in the response (top level DNS) and then that will tell you which one you have to talk to.

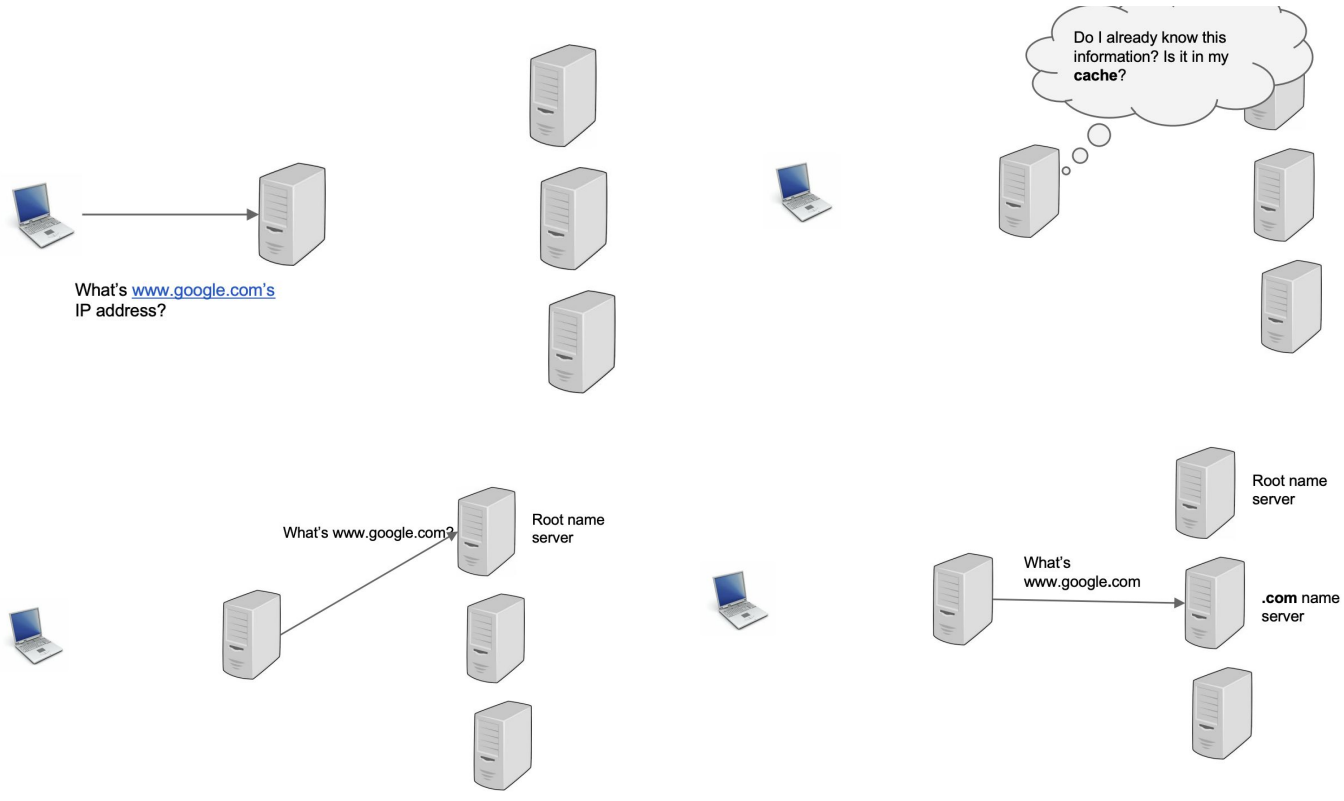


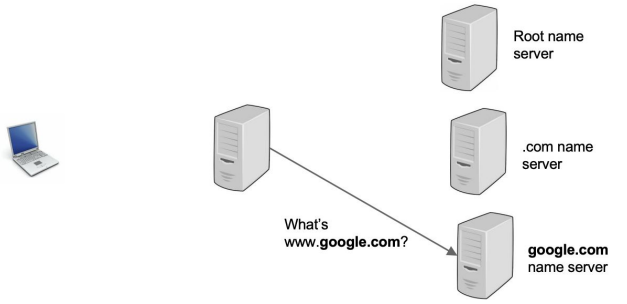
Recursive

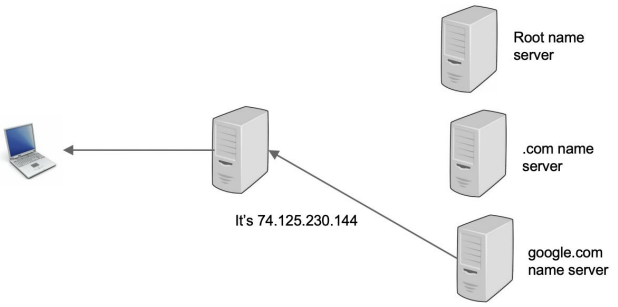


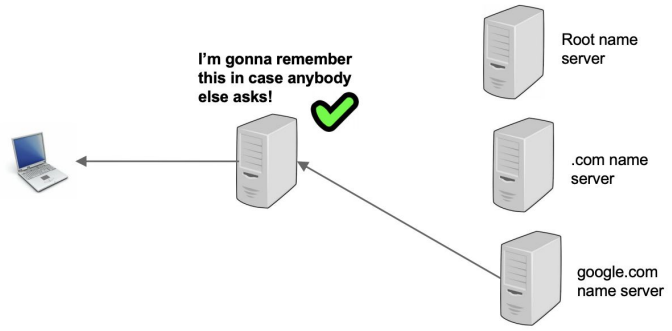
We tend to use iterative nowaday as it tends to be quicker.

To reduce overhead, we normally cache the IP associated to a name so that once I have found the mapping I don't have to look it up every time. This is critical for the performance of the WWW. This storage is done at a the local DNS level. Therefore, when other computers query for a certain name, I have that already in store.

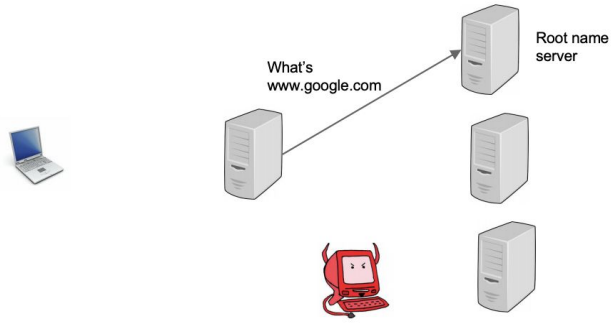


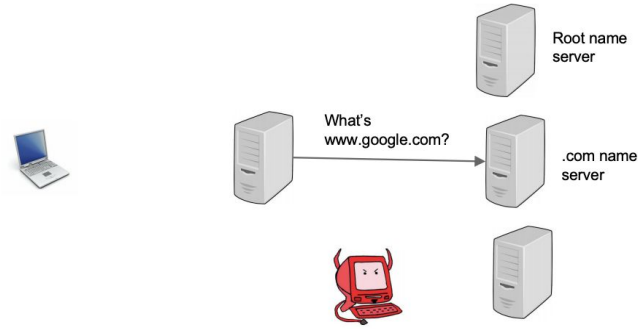


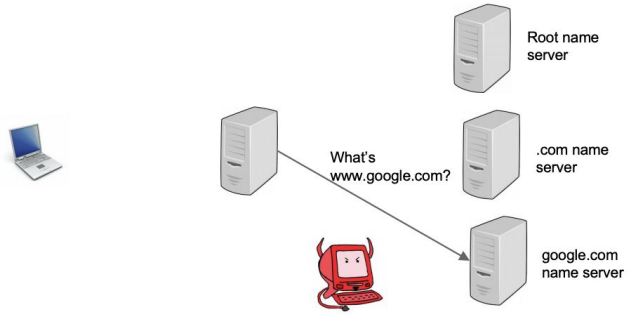


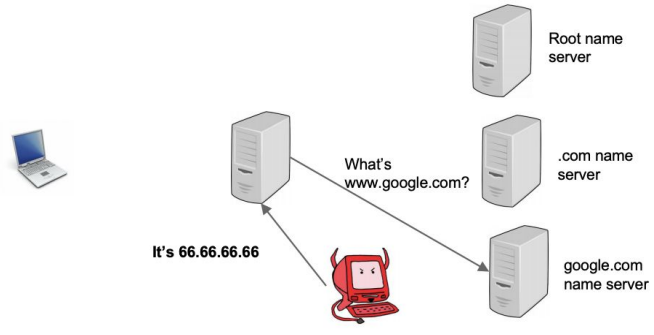


DNS attacks: poisoning

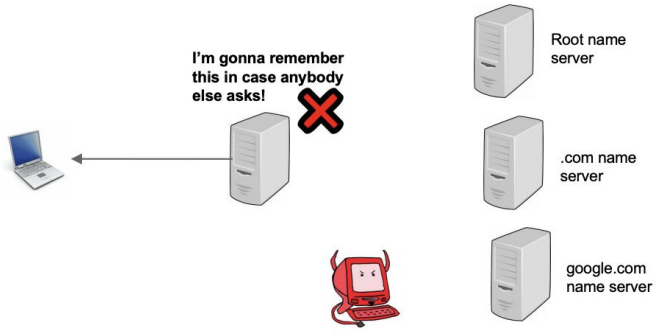


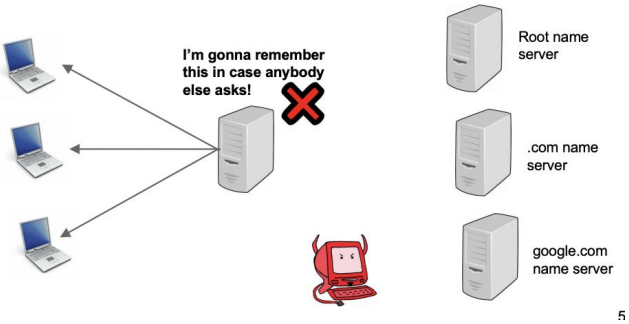




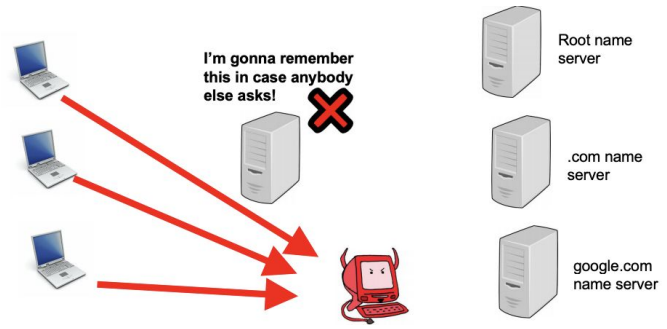


52





5



Now all the computers that turn to the poisoned DNS local router will be fed the malicious IP when they look for google.com. If this happened in KCL every student will get another site when we type google.com

If we do this, we can redirect any client to our server. DNS poisoning can also done for censorship by changing the IP the client requests rather than the one they reach.

How can we do DNS Poisoning

Note DNS works **on top of UDP (transport layer)**, **UDP is connectionless you do not expect a response once you send a package**. This means that you ask a question and you do not establish a connection with the DNS server, rather you just query, keep the query id and that's it.

1 Query ID (QID) attack

Whenever you make DNS query you store the query id that you created in the UDP request. Then you wait for a response with that specific query id.

As an attacker if we can guess that query id, we can send you whatever response we want. This query id is 16 bits, therefore it is difficult to guess but it is possible, especially if you can send a lot query responses at once. Bear in mind that you must send the response with the correct QID before the actual name server's response.

2 RSSet attack

DNS responses contain different Resource Record Sets or RRSets. Within these there is an "Additional section" where the nameserver can give additional information which may be useful for future lookups. I.e. include the IP address of the next DNS server that has to be asked in an iterative query. **Mallory** could abuse this feature and add another entry stating that the ip address of a **certainsite.com** is 121.123.123.123.

This was fixed with the Bailiwick check, which checked that when making queries for google.com you should not accept adding to the response information about another site like certainwebsite.com.

Although this was fixed, there's beena clever attack.

Dan Kaminsky's Attack (2008)

The attack works by making Alice send 1000s of queries (or Mallory sending) to non-existent domains like aaa.google.com or shaj.google.com. In this way Mallory can send answers to these queries with an additional record for google.com in each case.

Dan Kaminsky's attack

Normally DNS poisoning comes with website spoofing, to yield results. If we redirect people who are trying to go to google goes to linkedin then they know something is up. Instead, if we create a phishing attack on that site, we can obtain passwords and usernames to access a bank for instance.

Lab 4:
In our local machines at King's we are connected to the Ethernet, in this case we can see our private IP but not our public IP. Instead our public IP is King's public IP. That is hy when we executre $whois <our priv ip>, we cannot see any infomration. But how does a website know where to send a response? The anwer is a protocol called NAT which enables us to configure our network so that requests or responses coming from outside into our eco-system are assigned to a specific port on our machine, enabling us to communicate with the outter world.

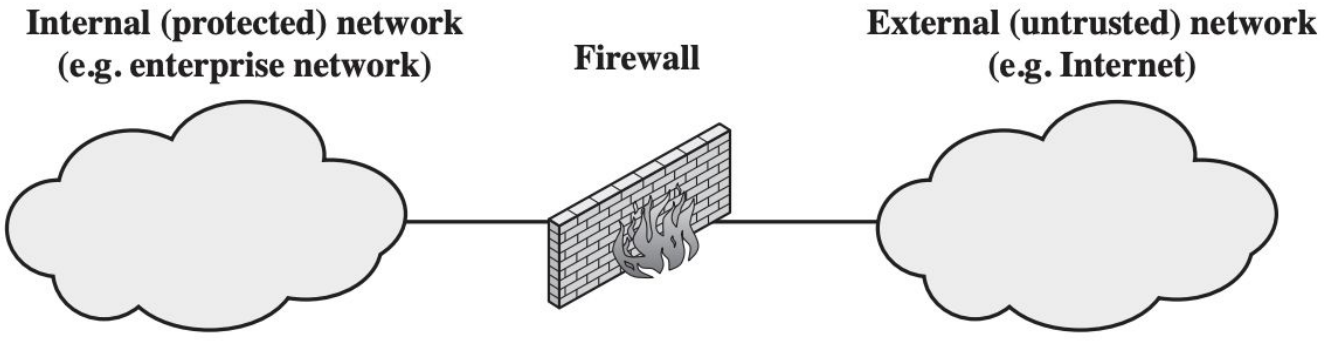
Lecture 5 Firewalls

Introduction

Firewall: the idea is to have one chokepoint in which everything passes through there so that you can control that.

Firewalls are used to bastion a network. They can be used to feed alerts to advanced Intrusion Detection Systems (IDS) and they can be used to implement Intrusion Prevention Systems (IPS).

What is a firewall?



Firewall: A firewall is a computer or a network security system that sits between your internal network and the rest of the network, and attempts to prevent bad things from happening (such as users sending company secrets outside, or outside people breaking into systems inside) without preventing good things from happening (such as employees accessing information available externally).

They are useful because:

- It interconnects networks with differing trust and imposes restrictions on network services: only authorized traffic is allowed.
- It can implement alarms for abnormal behavior (auditing and controlling access).

- Implements VPNs using IPSec.
- A firewall may be designed to operate as a filter at the level of IP packets, or may operate at a higher protocol layer.
- A firewall is as strong as the policy it follows. If the policy is weak, the firewall will be weak too.
- Provides NAT (network address translation) & usage monitoring

Must be immune to penetration.

Firewall security policy: specifies which traffic should be stopped and which allowed through a firewall in both directions.

Think of it as a border between two countries, the police have a policy to follow to allow you into the country or not.

Techniques used by Firewalls to control access and enforce security policy

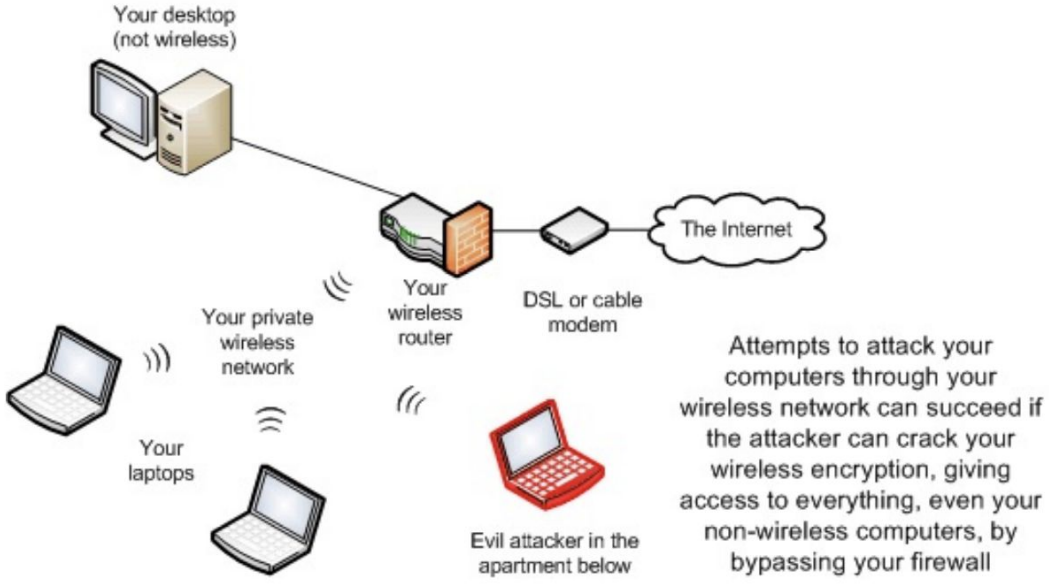
- **Service control:** Determines the types of Internet services that can be accessed, inbound or outbound.
- **Direction control**: Determines the direction in which particular service requests may be initiated and allowed to flow through the firewall.
- **User control:** Controls access to a service according to which user is attempting to access it.
- **Behavior control:** Controls how particular services are used.

Firewall Limitations

Firewalls cannot protect against attacks that:

- Attempts to bypass it
- Against malware imported via laptop, PDA, storage infected outside
- Against access via WLAN if improperly secured against external use
- Against internal threats
- Human error

If you walk into a company with a USB stick which is infected and plug it in, a firewall cannot do anything there.



If someone hacks into your private wireless network by breaking encryption and that router sits behind the firewall, then we effectively render the firewall useless.

Firewall types

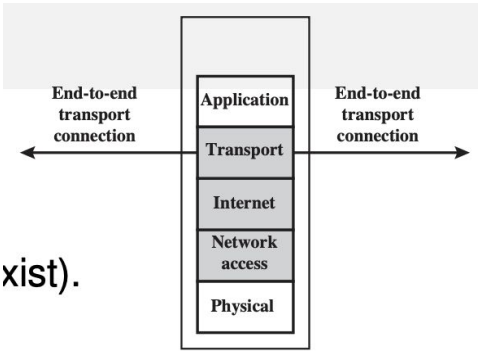
A firewall can operate as:

- a positive filter, allowing to pass only packets that meet specific criteria, or
- a negative filter, rejecting any packet that meets certain criteria.

Depending on the type of firewall, it may examine:

- it may examine one or more protocol headers in each packet,
- the payload of each packet, or
- the pattern generated by a sequence of packets.

Packet-filter firewall

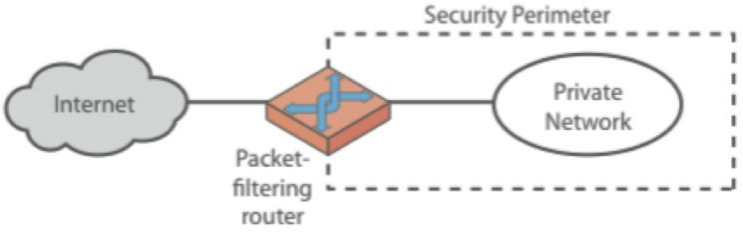


Packet-filter firewall: applies a set of rules to each incoming and outgoing IP packet and then forwards or discards the packet.
Simplest, fastest firewall component, they examine each IP packet (no context) and permit or deny according to rules. Hence restrict access to services (ports). Normally they look at the headers and make a decision.

xist).

Possible default policies:

- that not expressly permitted is prohibited,
- that not expressly prohibited is permitted -> this can be dangerous if you forget to block something that is a threat.



These firewalls filter packets using rules based on information contained in the network packet:

- **Source IP address:** The IP address of the system that originated the IP packet (e.g., 192.178.1.1).
- **Destination IP address:** The IP address of the system the IP packet is trying to reach (e.g., 192.168.1.2).
- **Source and destination transport-level address:** The transport-level (e.g., TCP or UDP) port number, which defines applications such as SNMP or TELNET.
- **IP protocol field**: Defines the transport protocol.
- **Interface**: For a firewall with three or more ports, which interface of the firewall the packet came from or which interface of the firewall the packet is destined for.

When we design a set of rules, we can block entire ranges of IPs or sub-networks using network masks:

Abbreviation	Address	Netmask
/27	32	255.255.255.224
/26	64	255.255.255.192
/25	128	255.255.255.128
192.168.1.0/24	256	255.255.255.0

You can choose if you want to accept these or not.

Example Rules

Packet-filter firewall is typically set up as a list of rules based on matches to fields in IP or TCP header. In each set, rules are applied top to bottom. The rule decides if the packet is discarded or allowed to pass through. If there is no match, then the default action is taken.

If there is a rule, then we use that rule, if there is no match, we continue down the rules and if we reach the bottom then we use the default action.

The default policy can be to allow or to discard

Default policies

Discard: what is not expressly permitted is prohibited.

- More conservative policy in which everything is blocked by default.
- Normally used by businesses and organizations.
- More annoying for users as they are more likely to hit the firewall.

action	src	port	dest	port	flags	comment
block	*	*	*	*	*	...

Forward: that which is not blocked is permitted.

- Easier to use for users are they do not normally encounter firewall.
- Reduced security and a lot of responsibility placed on network admin to block all malicious sites.
- Normally used by open institutions.

action	src	port	dest	port	flags	comment
allow	*	*	*	*	*	...

Depending on the type of router, filtering may be done:

- at input time (when it comes from the outer network in),
- at output time,
- or both;
- and by looking at control fields.

Granularity

Depending on the type of router, filtering may be done:

- at input time,
- at output time, or
- both; and
- by looking at control fields.

Rule Set A, B and C

- No direction applies
- Rule #A.1 blocks outbound packets to **theirhost** (SPIGOT)
- Rule #A.1 blocks inbound packets from **theirhost** (SPIGOT)

Rule Set A

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

Rule Set B

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

Rule Set C

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

Rule Set D

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

Rule Set 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

- * = ANY; default = discard

A Inbound mail is allowed (port 25 is for SMTP incoming), but only to a gateway host.
However, packets from a particular external host, SPIGOT, are blocked (e.g., because that host has a history of sending massive files in e-mail messages).

B This is an explicit statement of the default = discard policy.
All rule sets include this rule implicitly as the last rule.

Rule Set A

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

Rule Set B

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

Rule Set C

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

Rule Set D

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

Rule Set 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

* = ANY; default = discard

C This specifies that any inside host can send mail to the outside. A TCP packet with a destination port of 25 is routed to SMTP server on destination machine. The problem with this rule is that the use of port 25 for SMTP receipt is only a default; an outside machine could be configured to have some other application linked to port 25. As rule C is written, an attacker could gain access to internal machines by sending packets with a TCP source port number of 25.

D Achieves intended result that was not achieved in C. D exploits a feature of TCP connections: once a connection is set up, ACK flag of a TCP segment is set to acknowledge segments sent from other side. Thus, rule set D allows IP packets where source IP address is one of a list of designated internal hosts and the destination TCP port number is 25. It also allows incoming packets with a source port number of 25 that include ACK flag in TCP segment. We explicitly designate source and destination systems to define these rules explicitly.

E This rule set is one approach to handling FTP connections. With FTP, 2 TCP connections are used: control connection to set up file transfer, data connection for actual file transfer. Data connection uses a different port number assigned dynamically for transfer. Most servers, and hence most attack targets, use low-numbered ports; most outgoing calls tend to use a higher-numbered port, typically above 1023.

Thus, rule set E allows Packets that originate internally Reply packets to a connection initiated by an internal machine Packets destined for a high-numbered port on an internal machine This scheme requires that the systems be configured so that only the appropriate port numbers are in use. Rule set E points out difficulty in dealing with applications at packet filtering level. Another way to deal with FTP and similar applications is either stateful packet filters or an application-level gateway (see below)

Layer	Protocol
Application	DNS, NTP, FTP, SMTP, DNS, SSH, HTTP, HTTPS, DHCP, BGP
Presentation	TLS, SSL
Session	SOCKS
Transport	TCP, UDP, AH, ESP
Network	IP, ARP, ICMP
Data Link	-
Physical	-

Protocol	Port	Protocol	Port
SSH	22	IPsec	50, 51
HTTP	80	FTP	20,21
STMP	25	SSH	22

DNS	53	HTTPS (HTTP + SSL)	443
NTP	123	DHCP	67,68

Packet-filter firewall advantages & dadvantages

+ Very simple.
+ Typically are transparent to users
+ Very fast
- No examination of upper-layer data, hence no prevention of attacks that employ application-specific vulnerabilities or functions.
    - For example, cannot block specific application commands. If it allows a given application, all functions available within that application will be permitted.
- Most packet filter firewalls do not support advanced user authentication schemes (also due to lack of upper-layer functionality).
- Packet-filter firewalls are generally vulnerable to attacks and exploits that take advantage of problems within the TCP/IP specification and protocol stack, such as network layer address spoofing.
    - Many packet filter firewalls cannot detect a network packet in which the OSI Layer 3 addressing information has been altered.
- Due to the small number of variables used in decisions, packet filter firewalls are susceptible to security breaches caused by improper configurations.
    - It is easy to accidentally configure a packet filter firewall to allow traffic types, sources, and destinations that should be denied based on an organization's information security policy.
- Because of the limited information available to the firewall, the logging functionality present in packet filter firewalls is limited.

Attacks and countermeasures

IP address spoofing: The intruder transmits packets from the outside with a source IP address field containing an address of an internal host.

- Fake source address to be trusted (intruder transmits packets from the outside with internal host source IP address).
- Attacker hopes that use of a spoofed address will allow penetration of systems that employ simple source address security, in which packets from specific trusted internal hosts are accepted.

Countermeasure:

- Add filters on router to block such packets (discard packets with an inside source address if the packet arrives on an external interface).
- This countermeasure is often implemented at the router external to the firewall.

Source routing attacks:

- Attacker sets a route other than default (attacker specifies the route that a packet should take as it crosses the Internet, in the hopes that this will bypass security measures that do not analyze the source routing information).

Countermeasure:

- Block source routed packets (i.e., discard all packets that use this option).

Tiny fragment attacks: Split header info over several tiny packets.

- Attacker uses the IP fragmentation option to create extremely small fragments and force the TCP header information into a separate packet fragment. Attack designed to circumvent filtering rules that depend on TCP header information. Typically, a packet filter will make a filtering decision on the first fragment of a packet. All subsequent fragments of that packet are filtered out solely on basis that they are part of packet whose first fragment was rejected. Attacker hopes that the filtering firewall examines only the first fragment and that the remaining fragments are passed through.

Countermeasure:

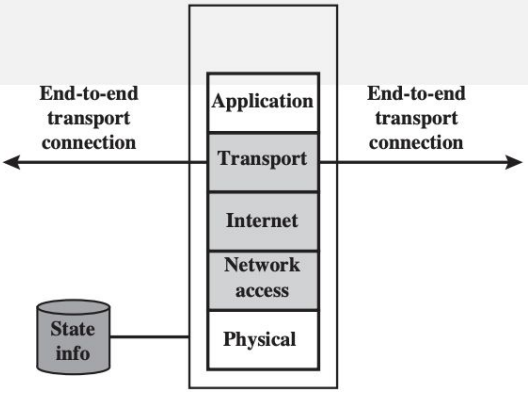
- Either discard or reassemble before check. A tiny fragment attack can be defeated by enforcing a rule that the first fragment of a packet must contain a predefined minimum amount of the transport header. If the first fragment is rejected, then the filter can remember the packet and discard all subsequent fragments.

Traditional packet filter makes filtering decisions on an individual packet basis, without considering any higher layer context i.e. matching return packets with outgoing flow. **Stateful packet filters** address this need: they examine each IP packet in context. Keep track of client-server sessions (record information about TCP connections), check each packet validly belongs to one. Hence are better able to detect bogus packets out of context. May even inspect limited application data.

Stateful packet filters

A stateful packet inspection firewall tightens up the rules for TCP traffic by creating a directory/table of outbound TCP connections. It creates an entry in a table for each established connection.

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.22.88.29	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
2122.22.123.32	2112	192.168.1.6	80	Established
210.922.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.21.22.12	1046	192.168.1.6	80	Established



Packet filter will now allow incoming traffic to high-numbered ports only for those packets that fit the profile of one of the entries in this directory.

A stateful packet inspection firewall reviews t**he same packet information as a packet filtering firewall, but also records information about TCP connections.**

- Some stateful firewalls also keep track of TCP sequence numbers to prevent attacks that depend on the sequence number, such as session hijacking.
- Some even inspect limited amounts of application data for some well-known protocols like FTP, IM and SIPS commands, in order to identify and track related connections.

IPTables is a popular tool used to filter network packets in Linux systems. It can be used as a stateful packet filter. There are two key concepts to understand how IPTables work: tables and chains

- **Tables**: general abstraction of the different chains: raw, **filter**, nat, mangle, and security.
- **Chains**: chains are inside of tables and contain defined rules to match against incoming packets ({INPUT, OUTPUT, FORWARD}). The action to perform is called **target** ({ACCEPT, REJECT}).

It is a kind of firewalls in Linux which allow us to establish policies.

Allow incoming SSH connections (22) and existing packets:

```
iptables -P DROP
iptables -A INPUT -p tcp --dport 22 -s $REMOTE_ADDR -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

1. Defining the default policy that drops every package.
2. Accept any package that comes with the tcp protocol with the destination port 22 and the source address is a remote address (this can be a mask, ie a range or one alone).
3. & 4. Accept any input and output connection that has been established or is related to the previous connections that I have accepted.

You can also set thresholds. I.e. Block a host that is attempting to connect via ssh more than x amount of times in 10 minutes.

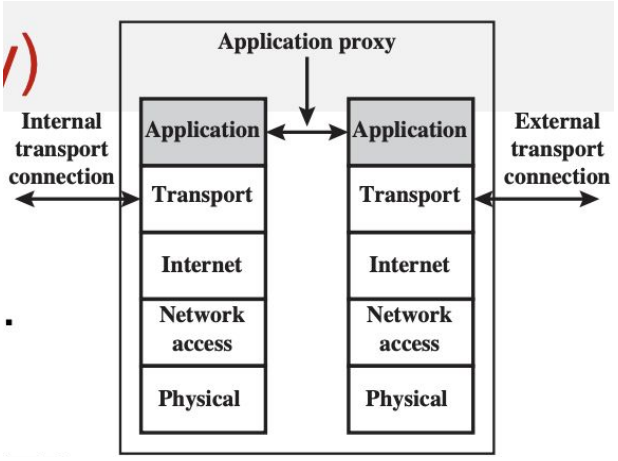
Application-level gateway (or proxy)

An **application-level gateway**, a.k.a. **application proxy**, acts as a relay of application-level traffic.

Is much more knowledgeable of what is going in, for example it may know about users and which users use which ports. From this info it can allow user1 to connect to such and such port but not user2. This may be used preventing dictionary attacks over ssh connections.

Has full access to protocol:

- User contacts gateway using a TCP/IP application, such as Telnet or FTP.
- Gateway asks user for name of remote host to be accessed.
- User responds and provides a valid user ID and authentication information (so that proxy can validate request as legal).
- Gateway contacts application on remote host and relays TCP segments containing application data between the two endpoints



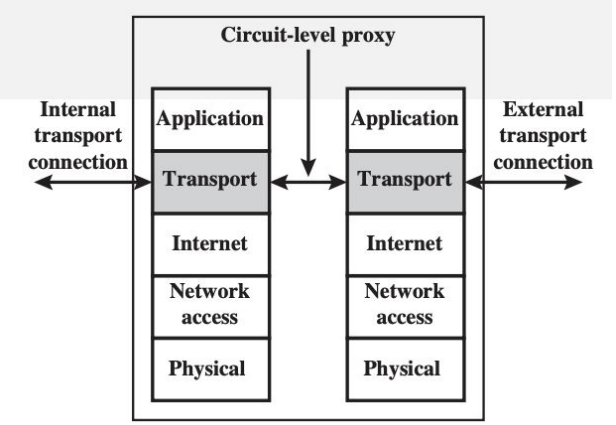
There kind of a default policy as if we do not specify the proxy code for a specific application, then the service is not supported and is not allowed through the firewall.

The Gateway can be configured to support only specific features of an application and not others, depending on the Network Administrator's criterion.

+ Are much more secure than packet filters because they better understand the context of the connection. Rather than trying to block packets which it deems malicious by looking at all the information at TCP and IP level it just focuses on scrutinizing what really matters, the applications it allows.
+ You get much more information about the current state of the computer thus allows better logging and better understanding of what is going on. Not only you know that there was a connection from IP1 to IP2 but you know by which user.

- There is additional processing overhead on each connection. The gateway must inspect all traffic going in and out.
- Need separate proxies for each service and some services may not support proxies.
- If application-level gateway does not implement the proxy code for a specific application, then the service is not supported and cannot be forwarded across the firewall.

Circuit-level gateway



A circuit-level gateway or circuit-level proxy can be a stand-alone system or a specialized function performed by an application-level gateway for certain applications.

This firewall only inspects packets at the transport level. It is normally implemented when you trust a user sending outbound packages. So long as you trust the IP address from which the data is being sent then you allow the traffic.

A circuit-level gateway does not permit an end-to-end TCP connection.

Circuit-level gateway sets up (and relays) two TCP connections:

- one between itself and a TCP user on an inner host,
- one between itself and a TCP user on an outside host

After you have established a connection, the firewall does not mind what happens next, it just relays the packets.

It's an in between between Application Level Firewalls and Filtering Firewalls.

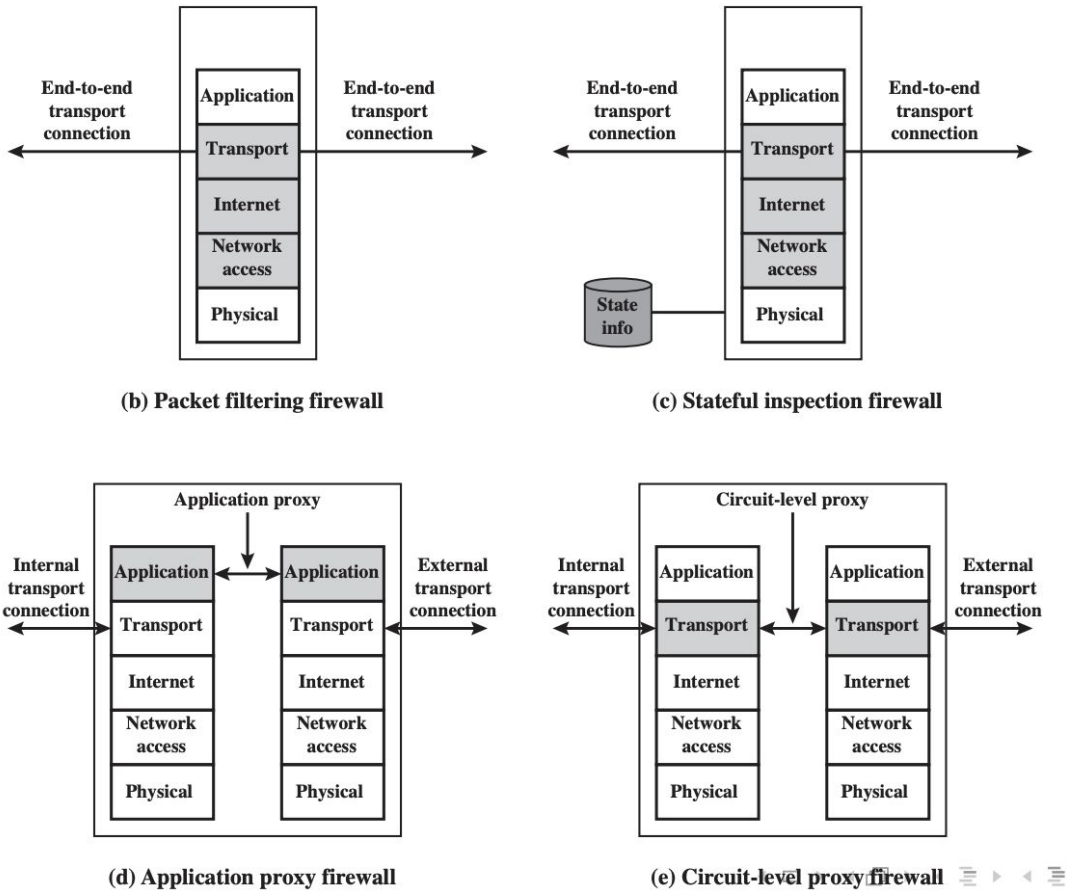
This firewall can be a stand-alone system or placed on a specialised function/section (to support) of an application-level gateway for certain application. It may incur the processing overhead of examining incoming application data for forbieed functions but can ignore outgoing data.

A circuit level gateway does not allow an end-to-end TCP connection. Rather it sits in between. It creates a TCP connection with the inner host and one with the outer host. It then relays TCP segments once the connection is established.

Its security comes from establishing which connections are allowed and which aren't.

Example implementation: SOCKet Secure (SOCKS)
SOCKS is an internet protocol that routes networks packets between a client and a server through a proxy server.

Recap



Firewall basing

Firewalls can be established **at different points** in the network and can be implemented in **hardware (Router, LAN switch, more efficient) or in software (UNIX, Linux)**.

Bastion host

Bastion host: It is a system that can be identified by the firewall administrator as a critical strong point in the network's security. A robust system that is typically placed close to the public internet. Between your internal network and the public network.

Typically, bastion host serves as a platform for an application-level or circuit-level gateway or mix of both (or provides externally accessible services).

It is potentially exposed to "hostile" elements, hence is secured to withstand this:

- Bastion host hardware platform executes a secure version of its operating system, making it a hardened system.
- Only services that network admin considers essential are installed on bastion host. These could include proxy applications for DNS, FTP, HTTP, SMTP

Bastion host may require additional authentication before a user is allowed access to proxy services.

Each proxy:

- May require its own authentication before granting access.
- Is configured to support only a subset of standard application's command set. So it cannot be reconfigured.
- It is configured to only allow access to specific host systems.
- Logs all traffic, connections and duration of these. It is useful for auditing which can be used to inspect, discover and terminate attacks.
- Normally is a small software packet designed to be easy to check for security flays (less than 1000 lines of code)
- Is independent of other proxies on the bastion host. (can be uninstalled/replaced without affecting others).
- New services are easy to install on a new bastion host.
- Runs as non privileged user in a private secured directory on the bastion host.

Host-based firewall

Host-based firewall: is a software module used to secure an individual host.

- Normally available in many operating systems but can be provided as an add-on package.
- They restrict the conventional flow of packages.
- Typically found on a server.

Advantages of server-based or workstation-based firewalls:

- Filtering rules can be tailored to environment (different servers can be implemented with different policies running different applications).
- Provide an added layer of protection when in conjunction with other stand-alone firewalls.
- A new type of server can be added to the network with its own firewall without having to change the main firewall.

Personal firewall

Personal firewall: controls traffic between a PC/workstation and the Internet or enterprise network.

- It typically is a software module on a computer and is less complex.
- The primary role is to allow the user to deny which type of incoming connections it wants to allow or deny.

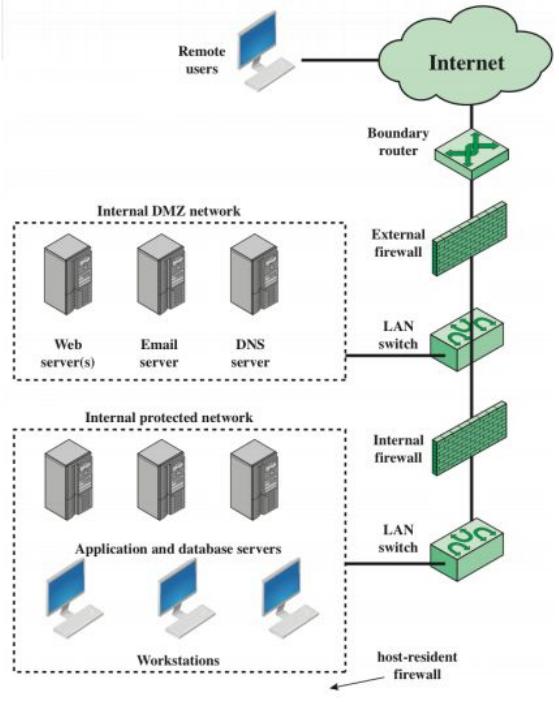
Firewall location and configuration

A firewall is positioned to provide a protective barrier between an external, potentially untrusted source of traffic and an internal network. Where it is located depends on the administrator. There are three ways:

Demilitarized Zone (DMZ) Networks

DMZ Network: is a network that sits between your connection and the outer world.

A firewall is positioned to provide a protective barrier between an external, potentially untrusted source of traffic and an internal network. One or more internal firewalls protect bulk of enterprise network.



Behind the external firewall you might place things which you expect to accept many connections for or that foster external connectivity like for instance Web, Email or DNS servers.

External firewall provides:

- Access control and protection for the DMZ systems.
- A basic level of protection for the rest of the network.

Internal firewall provides:

- More stringent filtering capability to protect enterprise network.
- Protects the enterprise network from attacks launched from the DMZ.
- Protects the DMZ from attacks from within the enterprise network.

A common practice is to place the DMZ on a different network interface on the external firewall from that used to access the internal networks.

Virtual Private Networks VPN

VPN: Consists of a set of computers that interconnect by means of a relatively insecure network which make use of encryption and special protocols to provide security.

It may be used to give the feeling that users which are not actually under the same network actually are under the same network.

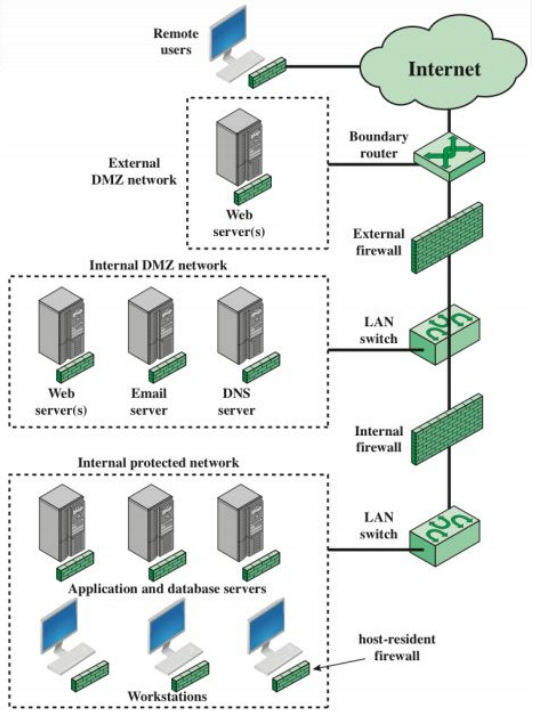
Uses encryption and authentication in the lower protocol layers to provide a secure connection through an otherwise insecure network, typically the Internet. Encryption may be performed by firewall software or by routers.

- Most common protocol mechanism used for encryption with this purpose is at the IP level and is known as IPsec

Distributed Firewalls

We can also have networks with distributed firewalls. These are stand-alone firewall devices and host-based firewalls working together under a central administrative control.

+ May be able to protect a certain workstation from attacking another workstation from within the network (internal attacks).
+ Provide better logging and analysis as all logs are tailored to each specific machine.
- Are much more difficult to configure.



With distributed firewalls you normally use both and internal and an external DMZ network. You make use of firewalls in a way that denotes how sensitive the information the machines behind it is.

Summary of firewalls locations and topologies

Host-resident firewall:

- Includes personal firewall software and firewall software on servers.
- Can be used alone or as part of an in-depth firewall deployment.

Screening router:

- A single router between internal and external networks with stateless or full packet filtering.
- Typical for small office/home office applications.

Single bastion inline:

- A single firewall device between an internal and external router.
- Firewall may implement stateful filters and/or application proxies.
- Typical configuration for small to medium-sized organizations.

Single bastion T:

- Similar to single bastion inline but has a third network interface on bastion to a DMZ where externally visible servers are placed.
- Typical configuration for small to medium-sized organizations.

Double bastion inline:

- DMZ is sandwiched between bastion firewalls.
- Common for large businesses and government organizations.

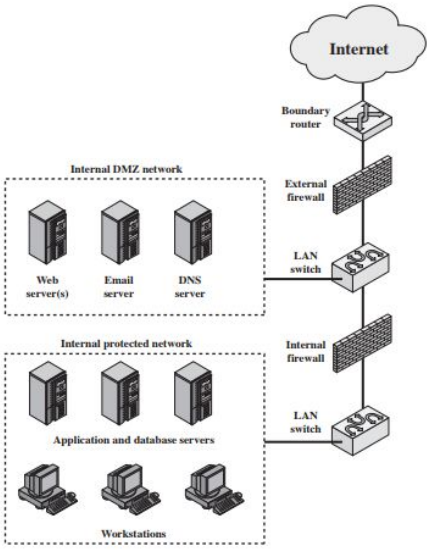
Double bastion T:

- DMZ is on a separate network interface on the bastion firewall.
- Common for large businesses and government organizations and may be required. For example, this configuration is required for Australian government use (Australian Government Information Technology Security Manual – ACSI33).

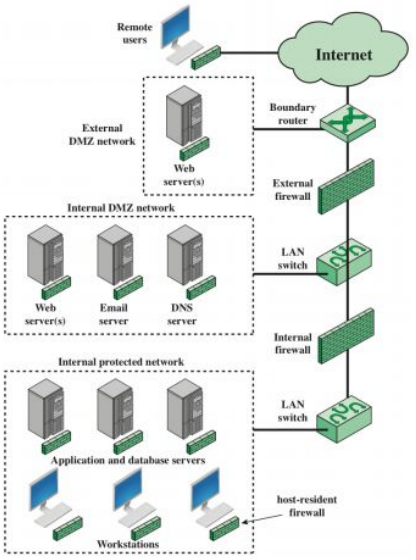
Distributed firewall configuration:

- Used by some large businesses and government organizations

What is a firewall? A firewall forms a barrier through which the traffic going in each direction must pass.
What is a policy? A firewall security policy dictates which traffic is authorized to pass in each direction.
How do firewalls operate? There are different types of firewalls. A firewall may be designed to operate as a filter at the level of IP packets, or may operate at a higher protocol layer.
Where do firewalls operate? Firewalls can be an effective means of protecting a local system or network of systems from network-based security threats



Double bastion inline



Distributed firewall configuration

L6&7 Intrusion Detection and Prevention Systems

Firewalls vs IDS

IDS and Firewall compliment each other. They provide in depth defence with multiple levels of security, it provides redundancy as protection eventually fails. But this comes at a cost.

These systems require a lot of attention as they need to be updated with new rules due to how quick attacks change. If a firm does not have someone actively monitoring these, it is not a good idea to have them as they are quite costly.

IDS: are a very good network monitoring system that have some very in depth rules that tell us if the packet is an attack or not. They typically know about the state of the network.

Definitions

Intrusion: Unwanted and unauthorized intentional access of computerized network resources
Intrusion detection: Detecting unauthorized use of a system or network Detecting attacks upon a system or network
Intrusion Detection System (IDS): A system that automates the detection of intrusions. They are to networks what Antiviruses are to systems but with packets instead of files.
Components:

- Sensors
- Alerts

Intrusion Prevention System (IPS): An IDS with an automated response. It actively tries to prevent and seek an attack.

- Shut down attacker connections
- Try to back-trace attacker
- Counter-attack

Intrusion Detection and Prevention System (IDPS): The same thing

IDPS Classification:

- **Timeliness** (moment of time in which the IDPS responds):
  - Real-time (online continuously running) [ideally], or
  - Non-real time (off-line, periodic, normally used in forensic analysis)
- **Response type:**
  - Passive: generates alerts
  - Active: blocks malicious traffic
- **State-dependency:**
  - Stateful analysis: requires more hardware an a more advanced way to gather and organise information.
  - Statesless analysis (look individually at each packet without understanding the TCP connection).
- **System type:**
  - Software
  - Hardware
- **Detection type:**

    - Reputation detection system: a system that is able to understand the identity of the user based on for instance the history. Decisions are based on how much they trust this host.
    - Misuse detection: those systems which are able to understand how an attack looks.
    - Anomaly detection: those systems which can identify things that are odd that don't normally occur.
- **Topology:**
    - Network IDS (NIDS)
    - Host IDS (HIDS): can put card in promiscuous mode and monitor the whole network.
    - Distributed IDS (DIDS): monitors the whole network with multiple probes [systems that send packages to other systems to test them] (network-and/or host-based)

Some kind of attacks to IDPS are preceded with dummy (Flood or DDoS, attacks which are easy to detect) attacks that aim to overload the analyst with a problem to attack some other part of the network. This will have an administrator block all connections but this may lead to someone having to deactivate the IDPS system as the rest of services still have to operate.

Reputation-based Detection IDS

Idea: detect if host communicating with someone with bad reputation. Based on public or private **blacklists** like:

- VirusTotal
- Malware Domain List
- PhishTank…

Our system is detecting if the packets that you are being sent depending on **who** sends them not **what** they send. So someone who has been bad in the past will be flagged over and over again.

Some of these may be OSINTs (Open Source Intelligence): which are platforms which collect data and release it publicly.

Misuse-based IDS

Misuse-based IDS: rely on models of malicious behavior (traditionally signatures). Identify matching entries in the event stream.

+ Generates few false alarms
+ Provides an explanation for alerts
+ It is fast
+ It is (more) resilient to evasion
- It detects only known attacks
- It needs continuous updating
- It is vulnerable to overstimulation attacks

Anomaly-based IDS

Anomaly-based: rely on models of normal behaviour. Identify anomalous entries in an event stream.

+ Detects previously-unknown attacks
    - Ie if you detect a process at 3am that should normally be done by an employee you know that is not right.
+ Does not need updating
- It is difficult to configure
- Assumes that anomalous means malicious

    - Users may start to behave in a different way and this might be picked up by our anomaly based detection system. This may lead to long configuration files.
- Generates many false alarms
- Does not provide a clear explanation of why this attack is happening.
- Resource-intensive

Indicators of Compromise (IOCs)

These are the different variables that may show signs of misuse.
Hostbased:

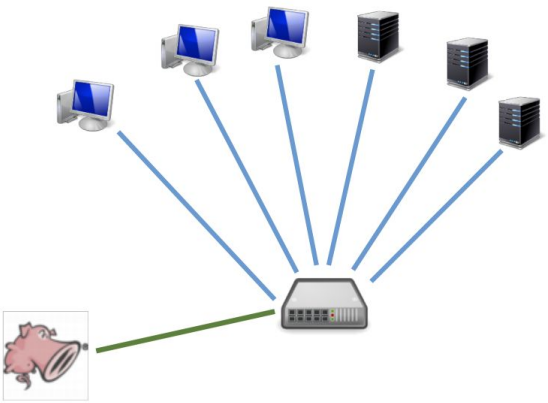
- Registry key
- Process name
- User account
- Directory path
- File name
- File hash
- Text string in file

Network-based

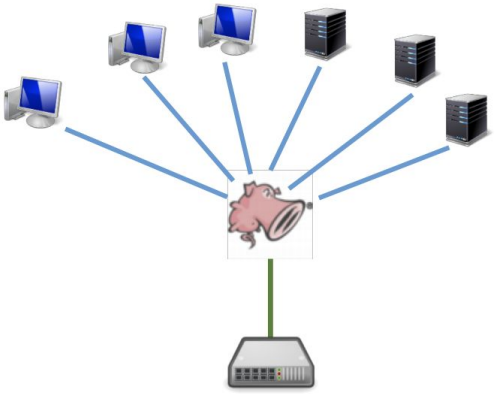
- IP address
- Port
- Protocol
- Domain name
- URL
- Downloaded file name or hash
- Text string in payload

Network IDs (NIDS)

Put card into promiscuous mode and sniff everything. They are normally located where the firewalls are located and aim to be a choke point. Typically placed in the DMZ.



Traditional deployment: the IDS is connected to a trunk port and can see all the network traffic
Can detect attacks, no countermeasures



Inline deployment: the IDS is deployed inline and analyses live traffic. De facto, it is an Intrusion Prevention System (IPS)
Attacks can now be prevented

Host IDS (HIDS)

Host IDS (HIDS): Looks at traffic on the local host: (they aim at protect a specific host)

- NIC not necessarily in promiscuous mode
- May also consider
    - Logs
    - System calls
    - Host activities
- IDS tuned for local services only, they do not mind about what occurs in different parts of the network.

Distributed IDs (DIDS)

Distributed IDS (DIDS): Multiple Sensors send events to a centralised Manager

- Sensors:
    - Collect events
- Combination of NIDS and/or HIDS
- Manager correlates collected events and informs if necessary
- Use a dedicated network or VPN for ProbeManager traffic

How an IDS works

1. Gathering the input information regarding what is happening in the network. This can be:
    - Application-specific information: data flow.
    - Host-specific information: local logs, system calls, file system changes.
    - Network-specific information: packets.

2. Look at the defined intrusion detection policy.
    - Known-good: Alert anything out of usual
        - Anomaly detection
    - Known-bad: Database with known attacks or attackers
        - Reputation-based detection
        - Signature-based detection
3. Generate a response on the intrusion
    - Active response
    - Passive response
4. Indepth Attack Analysis
    - Fine tune to understand the full picture.

Where to collect data:

- Alert Data: IDS alerts
- Log Data: Complete host logs
- Statistical Data: Network stats
- Session Data: 5-tuple flows
- Packet String Data: e.g. HTTP headers
- Full Packet Capture: PCAP files
    - Maybe only packet headers (64 bytes)

Tools to capture this data are:

- Packet capture: Tcpdump, dumpcap
- Session data: Netflow, IPFIX

- Session string: URLsnarf
- Logs: syslog

These systems can store terabytes (lots) of information, thus a balance is needed in the amount of detail and retention time of this detail necessary.

Network IDSs: Snort

Snort: is an open source **real-time misuse network** based intrusion detection system – based on libpcap. It allows us to do real time monitoring.

- It's a simple rule-based analysis engine
- It has pattern-matching capabilities to understand if any of the traffic is an attack given the rules.
- Support from the open source community though online rules are naive.
- Signatures
- Plugins, extensions

Snort Rules

They have two main components: header and options.

Header: defines matching packets:

- Rule's action / type (typically an alert)
- Protocol
- Source and destination IP addresses and netmasks
- Source and destination ports

Options: defines content matching and additional functions.

Example:

```
Alert tcp any any -> any any (content: "|90 90 90 90|";
msg: "shellcode detected";)
```

First is the action, then the protocol, the arrow indicates which address is the destination and which is the source, then you have the options, in this case send an alert where in the content of a package you see 90 90 90.

Snort has advanced lookup of packages which have been aggregated within a count, time or that have been tracked together. This is useful for a DDoS attack, whenever you see certain count of packages from the same source within a time period, show an alert.

Problems with Network IDSs

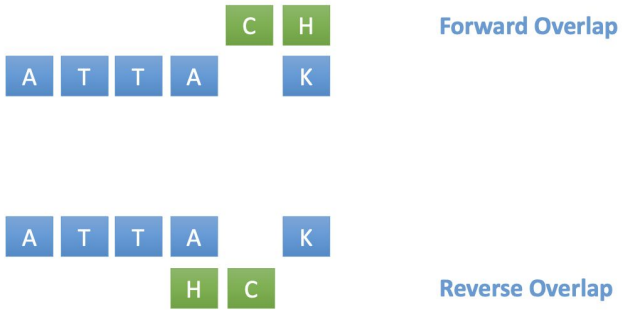
1. They require a **mirror port** on a switch to be able to observe the entire network traffic.
2. **Visibility problem** – if the network is partitioned (for example it has a DMZ) a separate IDS is required for each partition.
3. **Vulnerable to evasion techniques**
    - The sniffer logs also those packets that would be discarded by the operating system
    - Useless when encryption is used
    - The way packets are handled by the IDS and the end host might be different
    - Attackers could generate decoy attacks of perform very slow attacks

Insertion problem attack (evasion technique)

Since IDS want to work quickly, they might accept a packet that the end system will reject. For example, an attacker may send a number of packets out-of order. The IDS is looking for a specific sequence of packets to identify and attack. Because the packets are sent out of order and the IDS is looking for a specific order then the attacker is able to bypass this attack. Then the end system discards the malformed packet or puts them in order and the attack succeeds.

De-fragmentation behaviour attack

An attacker may send two TCP segments that overlap. Some systems may favour old data whilst others may favour new data. This may cause the IDS to be bypassed but the attack may still occur.



Other de-synchronization techniques

- The destination host may not accept TCP segments bearing certain options (ie 3 flags that don't belong together)
- The destination host might drop packets with old timestamps
- The destination host may not check sequence numbers on RST packets

Partial solutions

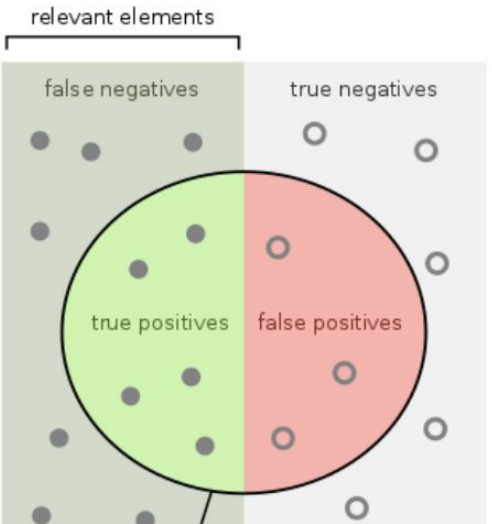
- Take into account the operating system of the target systems
- Take into account the installed services and their configuration
- Deploy a "normalizer" module that blocks malformed packets

This introduces a problem, namely that IDSs should be easily deployable and these countermeasures complicate this.

IDS Measuring Performance

IDS can make errors there are 4 different kinds of errors

		True Condition	
	TOTAL	Condition positive	Condition Negative
Predicted condition	Predicted condition positive	**True Positive (TP)**	**False Positive (FP)**
	Predicted condition negative	**False Negative (FN)**	**True Negative (TR)**



- Sensitivity, True positive rate (TPR), Recall, probability of detection
- Specificity (SPC), Selectivity, True negative rate (TNR)
- Accuracy = (TP + TN) / TOTAL
- F1 score = 1/(1/Recall + 1/Precision)/2
- ROC (Receiver Operating Characteristic Curve): Trade-offs between true positive (benefits) and false positive (costs)

Not relevant to exam

Key message: *you can message the performance in different ways. We have to make sure we do not detect packages that are normal as attacks and still able to detecting the bad ones.*

Honeypots

Honeypot: is a system that has been designed to be a trap to disclose/learn information about certain threats.

It is heavily used for information gathering purposes which can then aid the prevention and detection of attacks.

Traffic that arrives to the honeypot is deemed malicious by definition as only intruders would make it to the honeypot. They don't provide much security to the network but the information they gather can be critical to the overall security of the network. It is used to learn and understand how attacks work.
Normally, if a malicious user can penetrate the network it will look at other computers it can attack and these would normally be honeypots which will record as much information as possible when the attacker infiltrates.

Use Cases

- Antivirus creating signatures
- SPAM filters
- Used by ISP to understand if any systems are compromised.
- Assist law enforcement to track criminals
- Hunt and shut down botnets
- Malware collection and analysis

Deployment

- Typically located in the DMZ or outside the boundary router
    - Because these systems tend to have real vulnerabilities and you don't want to have a vulnerability in your internal network.
- Virtual Environment

Goal

- The honeypot looks as real as possible (good bait)
- Does good monitor
- Is relevant to the attack, namely it allows them to carry out their attack.

Classification

By level of interaction:

- High

- Low

By implementation:

- Virtual
- Physical (as attackers are really good at knowing if an environment have been virtualized, also more expensive).

By purpose:

- Production
- Research

Low Interaction Honeypots

- Emulate services, applications and Operating systems
- Cannot be exploited to get complete access to the honeypot (root access)
- The attacker is limited to the level of emulation that has been configured.

+ Low risk
+ Easy to deploy/maintain
- They capture limited information

High interaction Honeypots

A real system and/or network.

- Use real operating systems and applications, there is no emulation. The attacker has access to a real network and system.

+ Capture intensive information
- High risk and difficult to maintain: sometimes you have to restart it and make sure attackers have not breached your security.

Common Architectures

Honeypots are set up in highly bastioned networks so that if an attack is to occur, you are sure that it can be blocked before it reaches the real internal network. Moreover, normally every packet that reaches it is kept in order to be able to analyse it.

The typical phases are: data control/collection, data capture and data analysis.

Data Control

In this stage we aim to mitigate the risk of harming others, to ensure this we typically add a IDPS surrounding the honeypot. Normally, these systems have some kind of bandwidth throttling to ensure your honeypot is not attacking another services (i.e. via a DDoS attack). Moreover, at this stage honeypots also gather general statistics about the attacks. Time, frequency…

Data Capture

In a honey pot we want to keep all the data possible, that includes system activity, application activity and network activity.

Data Analysis

Normally a combination between Human-driven and Machine-driven analysis as parsing of the data has to occur.

Case Study: To Catch a Ratter

RAT (Remote Access Tool) called DarkComet

This type of malware gives you a useful graphical interface access to the compromised machine. This is typically commercialised in the black market. In this kind of attack there is always some hands-on work to control an attack. It is normally a human, not an automated attack.

In this study they used honeypots to study the motivations to use RAT.

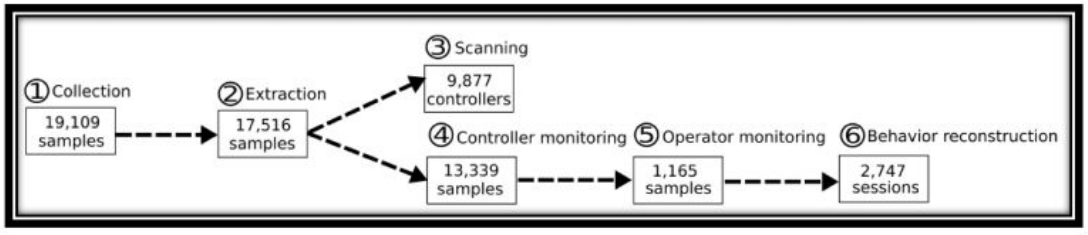
The typical RAT operations are:

- Capture audio from microphone
- Capture video from webcam
- Log keyboard input
- Browse files on machine

These are then used for extortion or blackmail or logging passwords to sell them in the black market.

This study aimed to target a very particular RAT called **DarkComet.** They attacked the tool by breaking the encryption between the packets that were sent between the honeypot and darkcomet, looked at different versions of the tool and very importantly at the campaign id. Given that most times access to other people's machines is sold/rented attackers must keep a record of each transaction to ensure they can properly offer the service.

Throughout the process they were able to collect information about 2747 sessions in which people were using RATs.



Methodology

Experiment 1	Experiment 2
20 Honey Pots ran concurrently	8 Honey Pots run concurrently
Identical Honey Pots with minimal cosmetic difference	• Unmodified Windows installation • PC Gamer (male) • Medical doctor (male) • U.S. political figure (female) • Academic researcher (male) • Bitcoin miner • College student (female) • Bank teller.
Honeypot Limitations • No webcam or microphone • No responses to attacker-initiated chat, communication • No keystrokes for keylogger • Network containment policy	

Given that these were run in virtual environments, sometimes this may giveaway the fact that an attacker is in a virtual environment. This is because when someone clicks on the webcam or microphone, they might not be able to gain access.

They looked at where these attacks came from. And found that it was mainly Turkey, Russia, and some from the US and France. Most of them tried to open an active (actually control) Remote Controlled session.

They realised that the first this people were trying to figure out was if there was a webcam and if they could record the user others aimed to open a remote controlled session or steal passwords.

Honeytokens

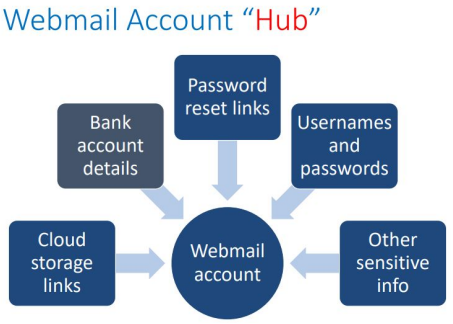
These is an example of another type of honeypot. Some of them may not be a full system. A honeypot can be an unused email address or a fake database entry.

Their value lied in their abuse but not their use. So what do attackers do with it. The first honeypot created was an email address. An example is when an unused email entry was placed into all the address books of all employees meaning that if that email inbox received a spam message, that meant that at some point there was a breach.

Case Study: Understanding The Use Of Leaked Account Credentials

In this case they created many honey accounts, fake gmail accounts (thousands). With the support from Google they were able to configure a system that would configure each account and be able to track all the activity. They then leaked the honeytokens and were able to monitor what malicious users were searching and how they were interacting with the user's account. They then recorded and analyzed all data that was found.

The aim of this research was to find what happens to your accounts after they have been compromised by cyber criminals. This was interesting as many users host all their other online accounts on a single email account.



The accounts were leaked in different places of the internet. Namely in the surface web, in typical search engines like Google, yahoo or even reddit. They also leaked them into the deep web and in the dark web.

They carried out multiple experiments to understand how tuser in each type of web exploited the different accounts.

Honey Accounts Leaked Via The Surface Web

- Created 100 Gmail honeypot accounts which with the help from google were able to modify timestamps and other metadata to make the accounts look more realistic.
- Populated them using the Enron corpus, these were real leaked account credentials which had been leaked in years prior.
- They mimicked the modus operandi of cybercriminals publishing them in underground forums, malware and paste sites.
- They then collected and analyzed data.

Honey Accounts Leaked Via The Dark Web

- Created 100 Gmail honeypot accounts which with the help from google were able to modify timestamps and other metadata to make the accounts look more realistic.
- Populated them using the Enron corpus, these were real leaked account credentials which had been leaked in years prior.
- Leaked account credentials via hidden paste sites, underground forums, and black market.
- They then collected and analyzed data.

Language Differentiation Study

Does language differentiation affect illegitimate accesses to webmail accounts?

The key takeaways of all research were that:

- Many more accesses to honey accounts originate from the Dark Web than the Surface Web
- Location (many people wanted to access UK accounts) and language differentiation affect accesses to honey accounts
- However, results do not necessarily reflect what happens in all compromised webmail or online accounts

Challenges of honeypots

- It is often difficult to find actionable deviations from what is normal behaviour or attacker behaviour.
- Intrusion events are not easy to find, they are not binary. Sometimes we might sense something which is noise or not an attack.
- Turning events into actionable alerts is difficult as the events are unexpected and change.
- There is a need to learn about new threats and this is done with honeypots and human analysis.

L8 Network Security Protocols

Network security preliminaries

Networks are logical and physical links which are connected to exchange bits.

Physically: a collection of “segments” that transmit bit streams.

- Examples: wire between two nodes or multi-access links like a LAN (e.g., Ethernet, token rings, packet radio networks).

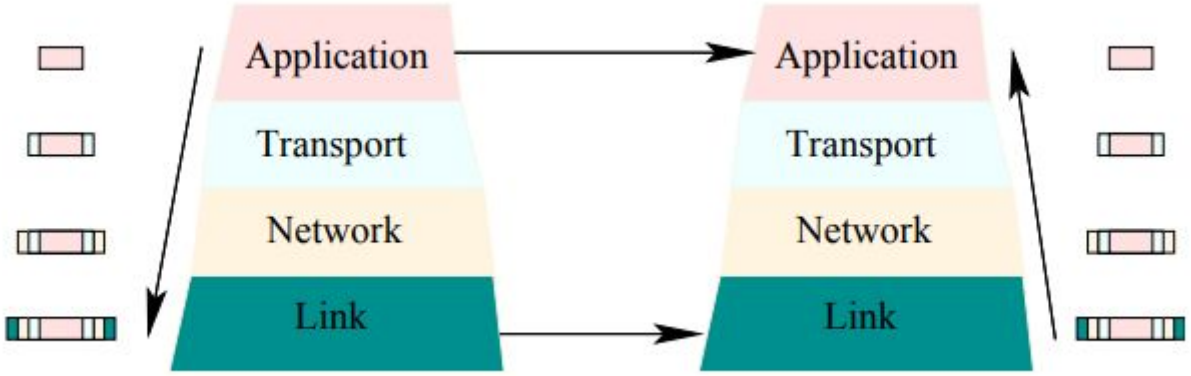
Logically: a communication medium between principals.

- Example: client communicates to server. A secure channel is yet another abstraction. Other abstractions may concern availability, privacy of communication partners, etc.

Communication in networks is layered.

- The application communication is built on
- Reliable transport between nodes which is built on
- Unreliable transport across links and switches (IP, Ethernet) which builds upon
- Packet transactions across single links

Application	Telnet, FTP, HTTP, RPC, SMTP, SET, ...
Transport/session	TCP (Transmission Control Protocol), UDP (User Datagram Protocol)
Network (Internet)	IP (Internet Protocol), ICMP (Internet Control Message Protocol), ...
Link (Interface)	Network interface & device drivers (IEEE 802.x, PPP, SLIP)



Each layer communicated with a layer in another system at the same level. Protocols typically just add a new header and a new payload.

The internet has not been built with security in mind. Every packet that we send typically makes 15+ hops and having to trust all of these is very difficult, especially since packet routes are dynamically assigned. Given that we cannot trust the means we must make sure our information is secure.

Which layer should we make secure?

Internet Protocol (IP): deliver data across a network.

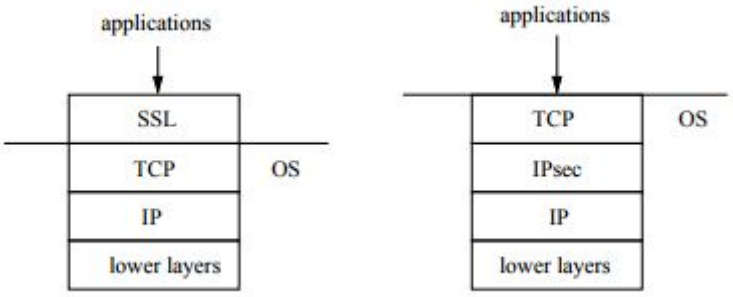
- Packet headers specify source and destination addresses.
- Protocol computes path and forwards packets over multiple links from source to destination.
- Current version is IPv4 (transition to IPv6 under way).

Transmission Control Protocol (TCP): establishes reliable communication between systems across a network.

- Reliable: either all data delivered without loss, duplication, or reordering, or the connection is terminated

Neither provide security: no authentication or confidentiality. Addresses can be faked. Payload can be read and modified.

Traditionally, security has been implemented at the application layer and the lower level layers have been typically managed by the operating system.



With SSL (or TLS) the operating system dealt with the lower level layers as normally and security was implemented by the applications. Nowadays with the implementation of IPsec, the OS is the one that deals with the security and the Applications and TCP remain unchanged.

Implementing security in the … layer:
Application (end-to-end):

+ No assumptions needed about security of protocols used, routers (under).
+ The decision to secure a communication or not can be done based on the user data.
- The applications must be designed to be security aware

Between the application and transport layer (SSL):

+ No modification to the OS, minimal changes to applications
- Poblems with interacting with TCP. SSL may reject data that TCP accepts. Then SSL must drop the connection and this opens to easy attacks (DDoS).

IPsec:

+ Transport layer security without modifying applications
- Only authenticates IP addresses, no user authentication
- More is possible, but requires changing API and applications.

IPsec

Background

IP security can be added to either the current version of the Internet Protocol (IPv4 or IPv6) by adding new headers.

By implementing security at the IP level, an organization can ensure secure networking not only for applications that have security mechanisms but also for the many security-ignorant applications.

IP-level security provides three functional areas:

- **Authentication:** allows you to ensure that a package is sent by a given entity. That the person sending that package is actually who it intends to be.
    - This is IP authentication not user authentication.
- **Confidentiality:** enables communicating nodes to encrypt messages to prevent eavesdropping by third parties.
- **Key Management:** concerned with the exchange of keys.

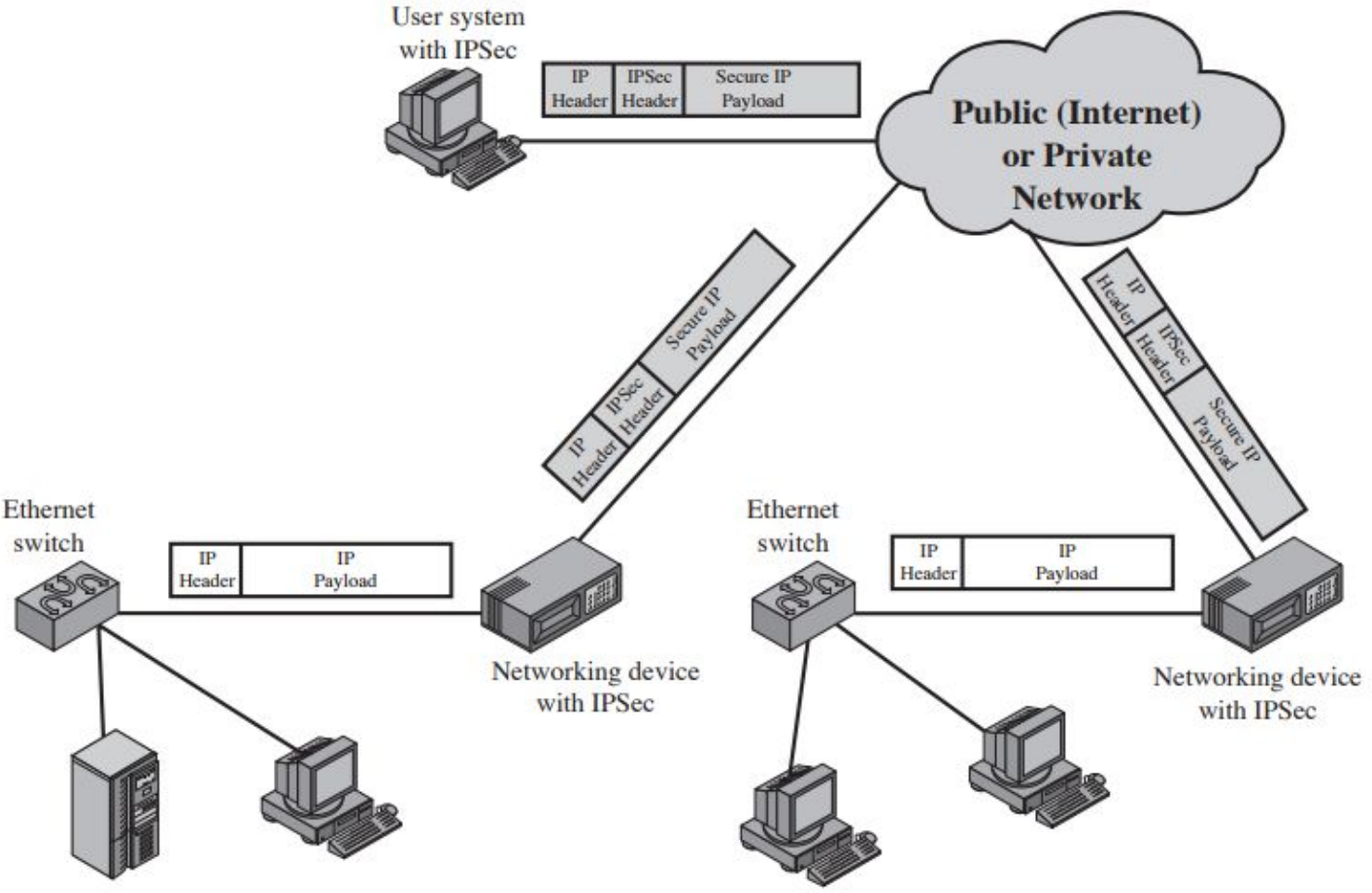
IPsec allows you to do some filtering and dropping some packages. It can be installed on top of an operating systems (for end-to-end security) or on top of gateways (firewalls or routers).

Applications of IPSec

Given that IPsec can secure authentication and encryption at IP level it allows distributed applications (including remote logon, client/server, e-mail, file transfer, Web access, …) to be secured.

- Secure branch office connectivity over the internet.
- Secure remote access over the internet.
- Establishing extranet and intranet connectivity with partners.
- Enhancing e-commerce security

An IPsec Scenario



- Although we have two LANs which seem insecure as they are operating with regular IP, our networking device (router or firewall) which operates IPsec is able to then apply IPsec and make our packet secure to be sent over the network.
- Secure transmission is also possible with individual users who connect to the WAN (remote worker). Such user workstations must implement the IPsec protocols to provide security.

Benefits of IPsec

+ When implemented in a firewall or router, IPsec provides strong security that can be applied to all traffic crossing the perimeter without traffic within a company or workgroup having to incur the overhead of security-related processing.
+ IPsec in a firewall is resistant to bypass if all traffic from outside must use IPsec or does not comply with the policy and firewall is only entrance from Internet into organization.
+ IPsec is below transport layer (TCP, UDP) and so is transparent to applications.

    + There is no need to change software on a user or server system when IPsec is implemented in the firewall or router.
    + Even if IPsec is implemented in end systems, upper-layer software, including applications, is not affected (if a user forgets to secure the data at application level that is not a problem or forgot https por example).
+ IPsec makes no changes to end-users.
    + There is no need to train users on security mechanisms, issue keying material on a per-user basis, or revoke keying material when users leave the organization.
+ IPsec can provide security for individual users if needed. This is useful for offsite workers and for setting up a secure virtual subnetwork within an organization for sensitive applications.

The IPsec standard

IPsec is an IETF Standard. Complex specification covering protocols for a variety of purposes:

- **Authentication Header (AH)**: protects the integrity and the authenticity of IP datagrams (but not their confidentiality).
- **Encapsulating Security Payload (ESP)**: protects confidentiality and optionally also integrity.
- **Internet Key Exchange Protocol (IKE)**: Provides a framework for policy (SA) negotiation and key management.

IPsec Services

IPsec provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services

	AH	ESP (encryption only)	ESP (encryption plus authentication)
Access control	✔	✔	✔
Connectionless integrity	✔		✔
Data origin authentication	✔		✔
Rejection of replayed packets	✔	✔	✔
Confidentiality		✔	✔
Limited traffic flow confidentiality		✔	✔

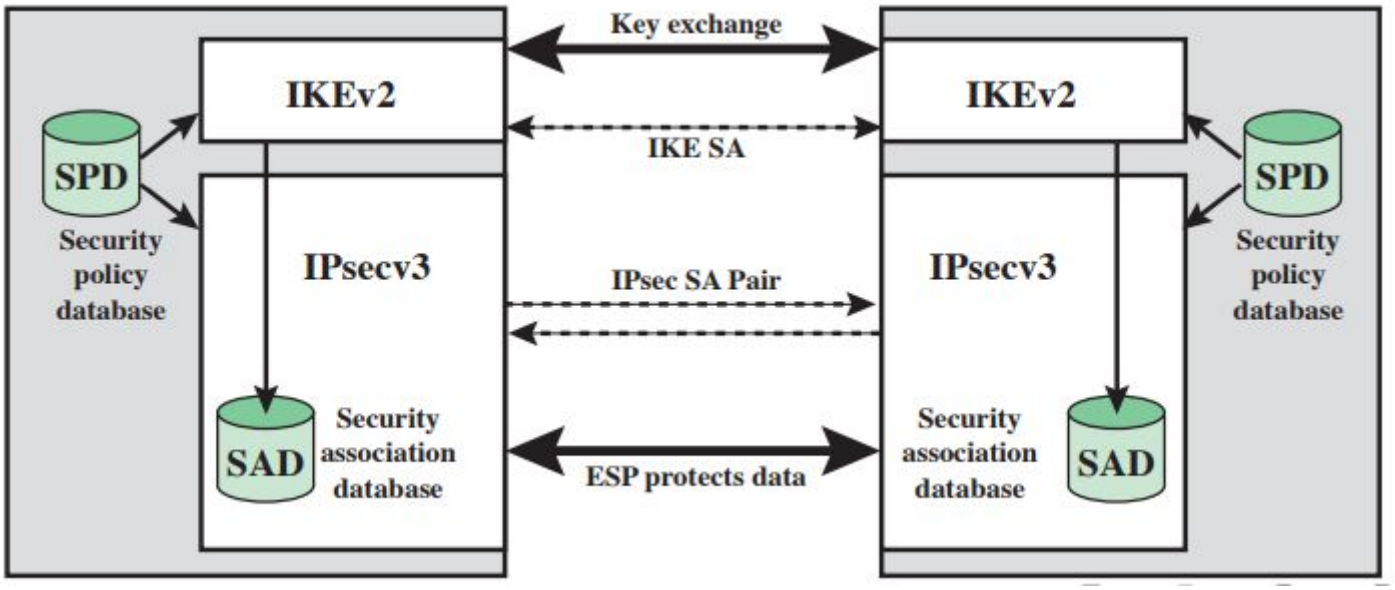
IP security policy

Fundamental to the operation of IPsec is the concept of a **security policy** applied to each IP packet that transits from a source to a destination.

Policies rely on a concept called **security associations** which are an agreement in the way to do cryptography.

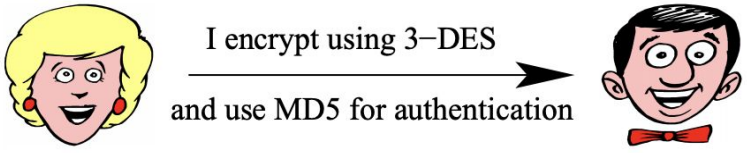
Policy is determined primarily by the interaction of two databases:

- the **security association database (SAD)** and
- the **security policy database (SPD)**.



Security Association Database (SAD)

Security Association Database (SAD): A security association is a one-way relationship between sender and receiver defining security services.



SA specifies things like: authentication algorithm (AH), encryption algorithm (ESP), keys, key lifetimes, lifetime of security association, protocol mode (tunnel or transport), …

SA is uniquely identified by three parameters:

- **Security Parameters Index (SPI):** a bit string assigned to this SA and having local significance only. SPI is carried in AH and ESP headers to enable receiving system to select SA under which a received packet will be processed.
- **IP Destination Address:** address of destination endpoint of SA (may be an end-user system or a network system such as a firewall or router).
- **Security Protocol Identifier:** a field from the outer IP header that indicates whether SA is an AH or ESP SA.

Security Policy Database (SPD)

Security Policy Database (SPD): means by which IP traffic is related to specific SAs (or no SA in the case of traffic allowed to bypass IPsec).

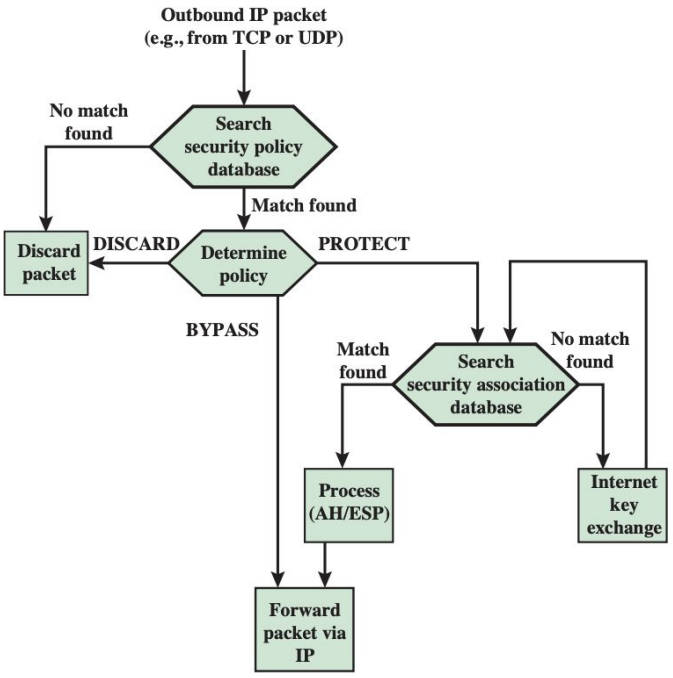
- In its simplest form, an SPD contains entries, each of which defines a subset of IP traffic and points to an SA for that traffic.
- In more complex environments, there may be multiple entries that potentially relate to a single SA or multiple SAs associated with a single SPD entry.

IP traffic processing

IPsec is executed in a packet-by-packet basis.

When IPsec is implemented:

- each outbound IP packet is processed by the IPsec logic before transmission, and
- each inbound packet is processed by the IPsec logic after reception and before passing the packet contents on to the next higher layer (e.g., TCP or UDP).



IP traffic processing: outbound

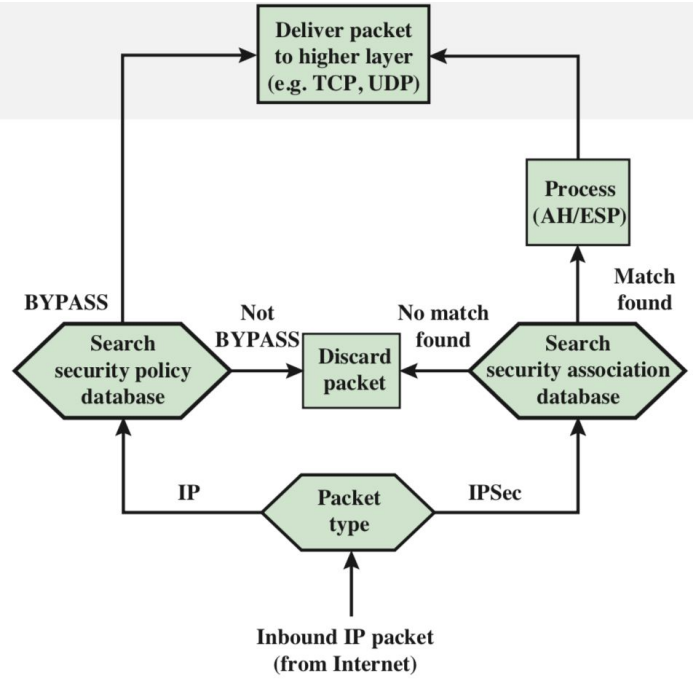
A block of data from a higher layer (e.g., TCP), is passed down to IP layer and an IP packet is formed, consisting of an IP header and an IP body.
Then the following steps occur:

1. IPsec searches Security Policy Database (SPD) for a match to this packet.
2. **If** no match is found, packet is discarded and an error message is generated.
3. **Else If** a match is found, further processing is determined by the first matching entry in the SPD, if the policy for this packet is:
    a. DISCARD, then the packet is discarded;
    b. BYPASS, then there is no further IPsec processing (the packet is forwarded to the network for transmission);
    c. PROTECT, then SAD is searched for a matching entry

If no entry is found, the **IKE protocol** is invoked to create an **SA** with the appropriate keys and an entry is made in the **SAD**.

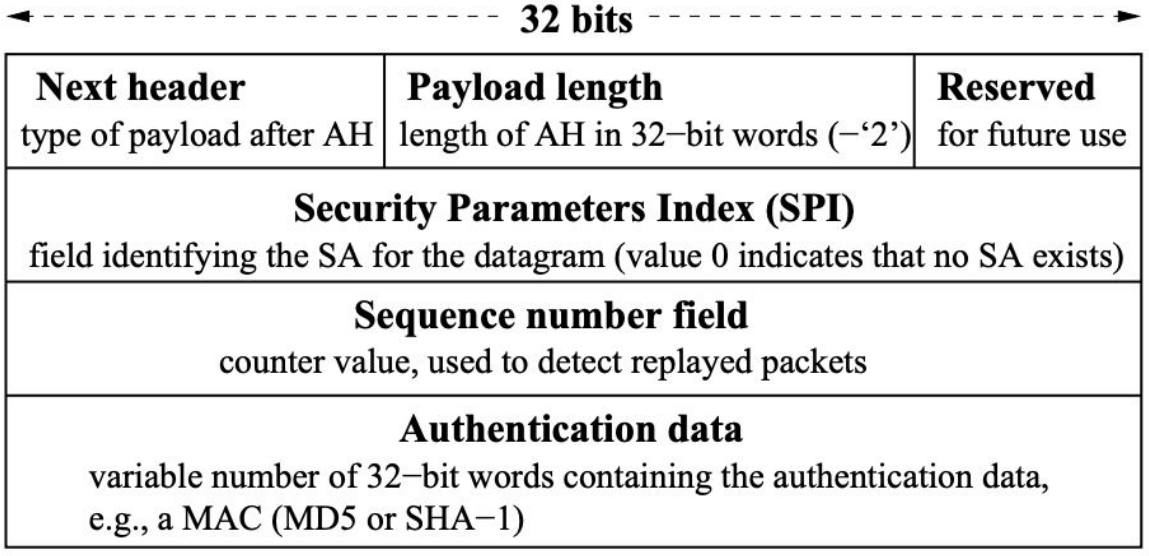
Matching entry in SAD determines processing for this packet. Either encryption, authentication, or both can be performed, and either transport or tunnel mode can be used. The packet is then forwarded to the network for transmission.

IP traffic processing: inbound

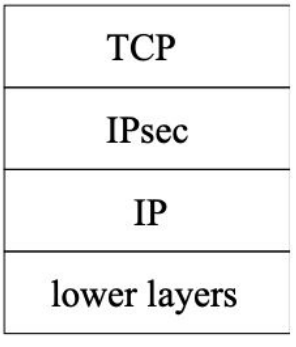


1. IPsec determines whether this is an unsecured IP packet or one that has ESP or AH headers/trailers, by examining the IP Protocol field (IPv4) or Next Header field (IPv6).
2. **If packet is unsecured**, IPsec searches SPD for a match to this packet. If the first matching entry has a policy of
    a. BYPASS, IP header is processed and stripped off and packet body is delivered to next higher layer, such as TCP
    b. PROTECT or DISCARD, or if there is no matching entry, packet is discarded.
3. **Else if the packet is secured**, IPsec searches the SAD.
    a. If no match is found, then packet is discarded.
    b. Else IPsec applies appropriate ESP or AH processing, and then IP header is processed and stripped off and packet body is delivered to next higher layer, such as TCP.

Authentication Header



Extra header between layers 3 and 4 (IP and TCP) providing the destination enough information to identify SA. AH guarantees integrity only, but also protects part of IP header.

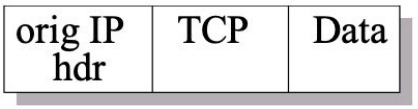


Sequence number: is initialized to zero and incremented by sender for each packet. Receiver stores incoming packets in a sliding window (default size 64) to order and sort out duplicates. (IP does not guarantee delivery or order.)

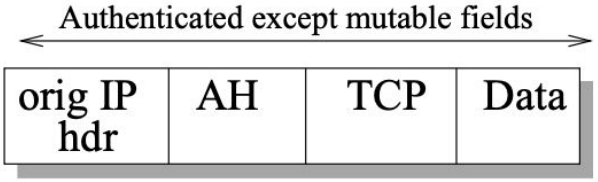
Authentication Data: is normally an encrypted hash or MAC that allows us to verify the integrity of the package.

Authentication Header modes

When we use the Authentication header in a certain mode by default it is like in the the picture on the right:

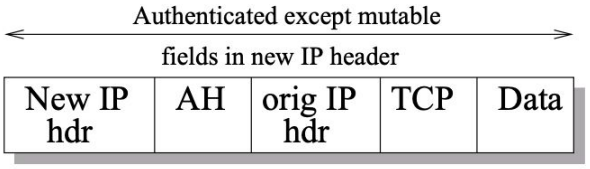


Transport mode:
Enables normal communication



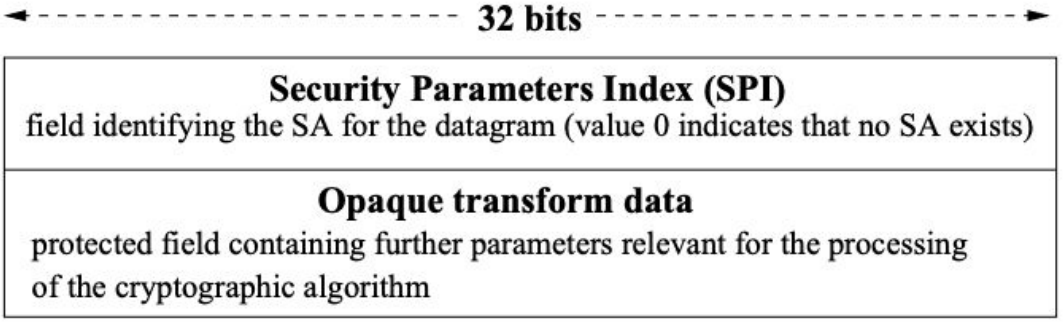
- AH inserted after IP header, before IP payload
- MAC taken of entire packet (except for mutable fields) to compute integrity.
- Provides end-to-end protection between IPsec-enabled systems
- All systems using transport mode have to have IPsec enabled.

Tunnel mode:
Allows us to establish connections like VPN allow us.



- Entire original packet authenticated; new outer IP header.
- Inner header carries ultimate source/destination address.
- New outer header also protected (except mutable fields) and may contain different IP addresses, e.g., firewalls or security gateways.

Encapsulating Security Payload (ESP)

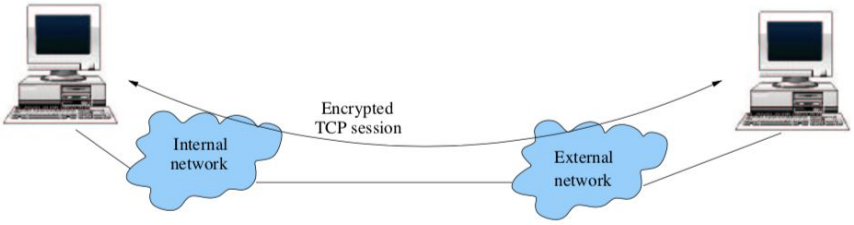


ESP can be used to:

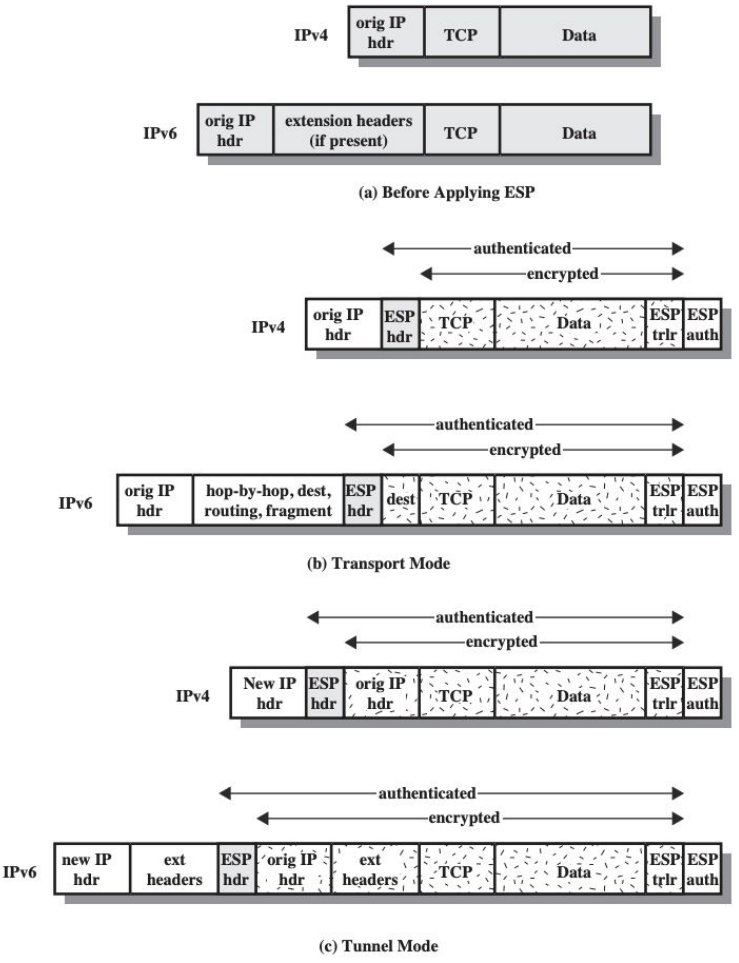
- Can be used to provide confidentiality, data origin authentication, connectionless integrity, an anti-replay service, and (limited) traffic flow confidentiality.
- Set of services provided depends on options selected at the time of establishment of the SA and on the location of the implementation in a network topology.
- Header specifies encryption and optional authentication

ESP applications: Tunnel mode

Transport mode: provides end-to-end encryption between hosts supporting IPsec.



Encrypts only the data portion (payload) of each packet , but leaves the header untouched.



IPv4:
ESP header is inserted into IP packet immediately prior to transport-layer header (e.g., TCP, UDP, ICMP), and an ESP trailer (Padding, Pad Length, and Next Header fields) is placed after IP packet.

- If authentication is selected, ESP Authentication Data field is added after ESP trailer.
- Entire transport-level segment plus ESP trailer are encrypted. Authentication covers all of ciphertext plus ESP header.

IPv6:
ESP is viewed as an end-to-end payload (i.e., it is not examined or processed by intermediate routers).

- Thus, ESP header appears after IPv6 base header and hop-by-hop, routing, and fragment extension headers.
- Destination options extension header could appear before or after ESP header, depending on semantics desired. This is because IPv6 allows us to do hop-by-hop destination routing.
- For IPv6, encryption covers entire transport-level segment plus ESP trailer plus destination options extension header if it occurs after ESP header.
- Again, authentication covers ciphertext plus ESP header.

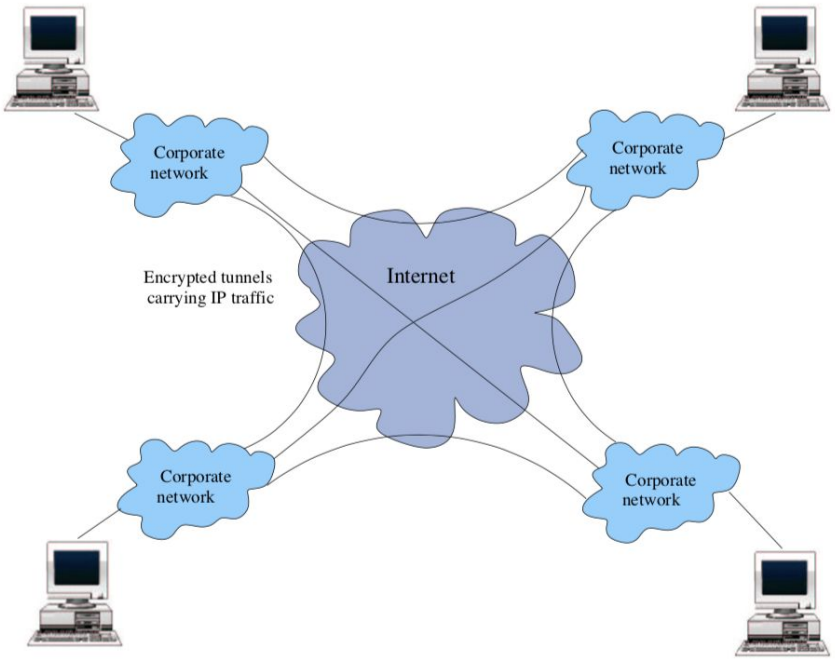
Summary

1. At the source, the block of data consisting of the ESP trailer plus the entire transport-layer segment is encrypted and the plaintext of this block is replaced by the ciphertext form to create an IP packet for transmission. Authentication is added if the option is selected.
2. Packet is then routed to destination. Each intermediate router needs to examine and process the IP header plus any plaintext IP extension header but does not need to examine the ciphertext.
3. Destination node examines and processes IP header plus any plaintext IP extension hedears. Then depending of SPI in the ESP header, the destination node decrypts the remainder of the packet to the plaintext of the transport-layer segment.

Transport mode operation provides confidentiality for any application that uses it, thus avoiding need to implement confidentiality in every individual application. One drawback to this mode is that it is possible to do traffic analysis on the transmitted packets.

ESP applications: Tunnel mode

Tunnel mode: can be used to set up a VPN.



- Hosts on internal networks use Internet to transport data but do not interact with other internet-based hosts.
- By terminating tunnels at security gateway to each internal network, configurations allow hosts to avoid implementing a security capability.

Tunnel mode ESP is used to encrypt an entire IP packet.

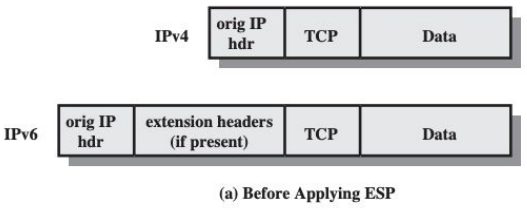
- ESP header is prefixed to packet,
- packet plus ESP trailer is encrypted.

This method can be used to counter traffic analysis.

As IP header contains destination address and possibly source routing directives and hop-by-hop option information, it is not possible simply to transmit encrypted IP packet prefixed by ESP header.

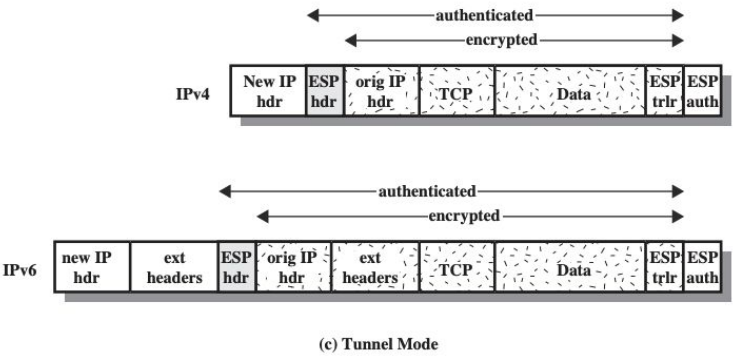
- Intermediate routers would be unable to process such a packet
- Thus: encapsulate entire block (ESP header plus Ciphertext plus Authentication Data, if present) with a new IP header that will contain sufficient information for routing but not for traffic analysis

Whereas transport mode is suitable for protecting connections between hosts that support ESP feature, tunnel mode is useful in a configuration that includes a firewall or other sort of security gateway that protects a trusted network from external networks.



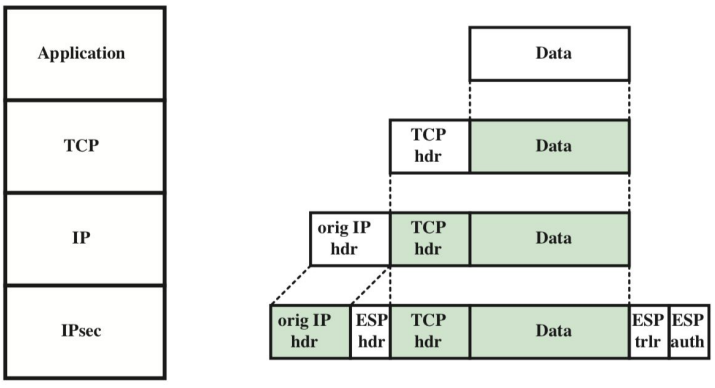
In this latter case, encryption occurs only between an external host and security gateway or between two security gateways. This relieves hosts on internal network of processing burden of encryption and simplifies key distribution task by reducing number of needed keys.

Further, it does not allow traffic analysis based on ultimate destination.
The transfer of transport-layer segment from external to internal host occurs in the following way:

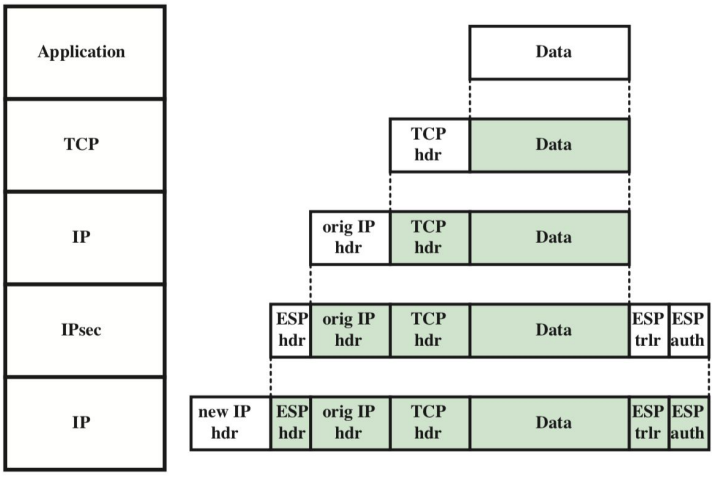


(c) Tunnel Mode

1. Source prepares an inner IP packet with a destination address of the target internal host.
    a. This packet is prefixed by an ESP header; then packet and ESP trailer are encrypted and Authentication Data may be added.
    b. Consider a case in which an external host wishes to communicate with a host on an internal network protected by a firewall, and in which ESP is implemented in external host and the firewalls.
    c. Outer IP packet: resulting block is encapsulated with a new IP header (base header plus optional extensions such as routing and hop-by-hop options for IPv6) whose destination address is the firewall.
2. Outer packet is routed to destination firewall. Each intermediate router needs to examine and process outer IP header plus any outer IP extension headers but does not need to examine ciphertext.
3. Destination firewall examines and processes outer IP header plus any outer IP extension headers. Then, on basis of the SPI in ESP header, destination node decrypts the remainder of packet to recover the plaintext inner IP packet. This packet is then transmitted in internal network.
4. Inner packet is routed through zero or more routers in internal network to destination host.



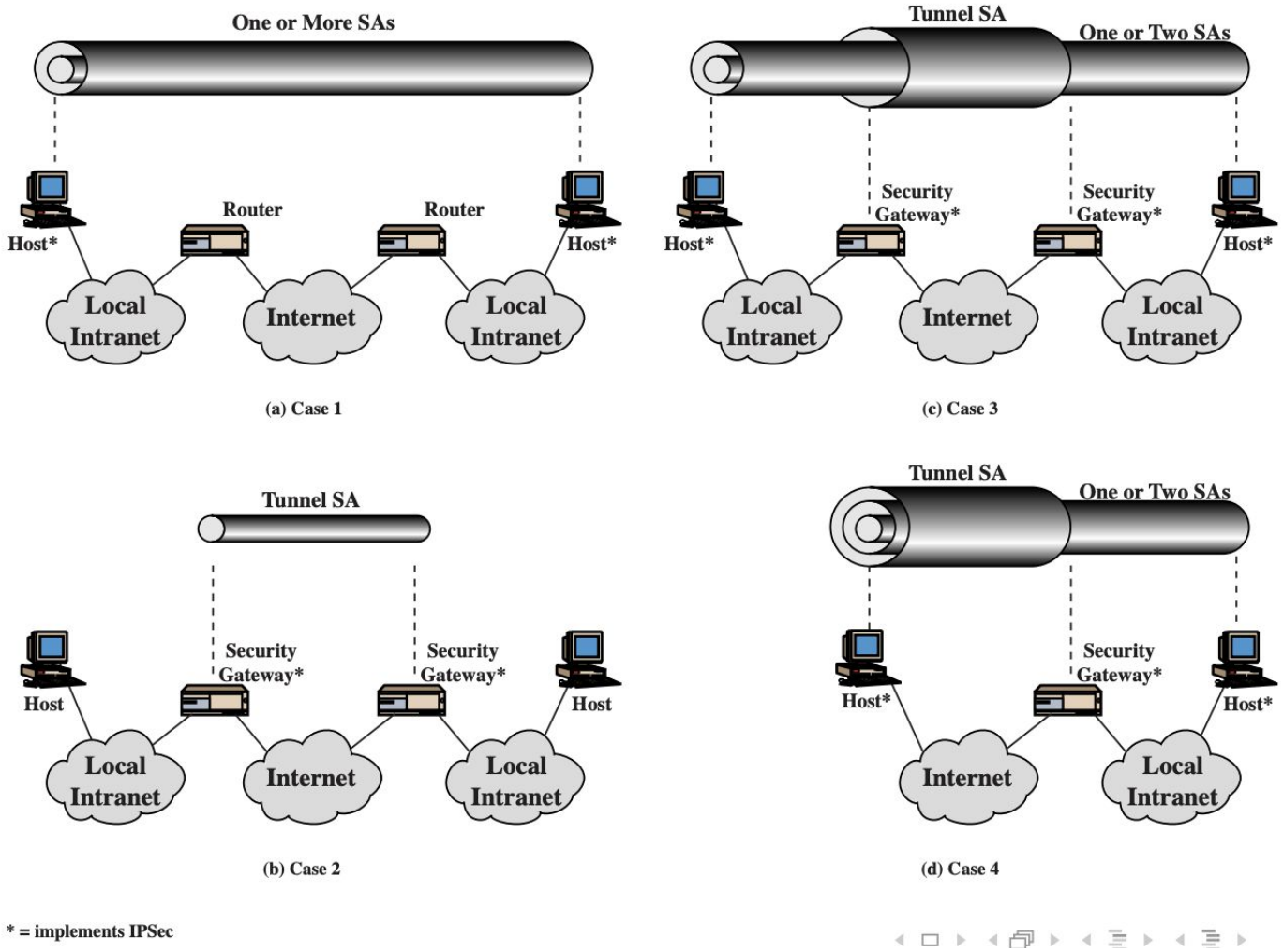
(a) Transport mode



Combining Security Associations

An individual SA can implement either the AH or ESP protocol but not both. Sometimes a particular traffic flow will call for the services provided by both AH and ESP. Further, a particular traffic flow may require IPsec services between hosts and, for that same flow, separate services between security gateways, such as firewalls. In all of these cases, multiple SAs must be employed for the same traffic flow to achieve the desired IPsec services. The term **security association bundle** refers to a sequence of SAs through which traffic must be processed to provide a desired set of IPsec services. The SAs in a bundle may terminate at different endpoints or at the same endpoints.

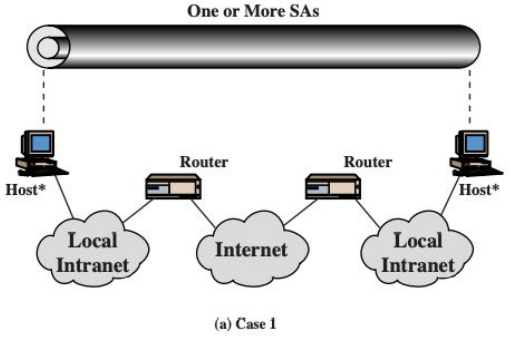
4 Examples



* Lower part of each case represents physical connectivity of elements.
* Upper part represents logical connectivity via one or more nested SAs.
* Each SA can be either AH or ESP.
* For host-to-host SAs, mode may be either transport or tunnel; otherwise it must be tunnel mode

Case 1

A way in which we combine security associations between two nodes that are in transport mode.



All security is provided between end systems that implement IPsec. For any two end systems to communicate via an SA, they must share appropriate secret keys

Among the possible combinations are

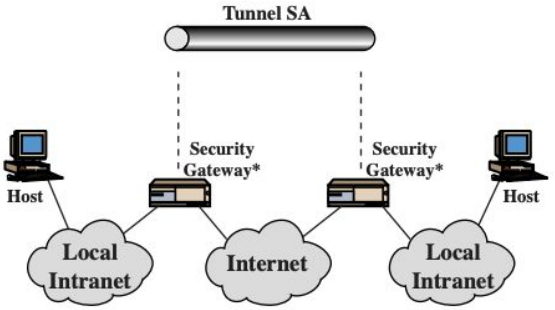
- AH in transport mode.
- ESP in transport mode.
- ESP followed by AH in transport mode (an ESPSA inside an AHSA)
- Any one of (i), (ii) or (iii) before inside an AH or ESP in tunnel mode.

As we have discussed, these various combinations can be used to support authentication, encryption, authentication before encryption, and authentication after encryption

Case 2

Security is provided only between gateways (routers, firewalls, etc.) and no hosts implement IPsec.

- This case illustrates simple virtual private network support.
- The security architecture document specifies that only a single tunnel SA is needed for this case.
- Tunnel could support AH, ESP, or ESP with authentication option. Nested tunnels are not required, because IPsec services apply to entire inner packet.

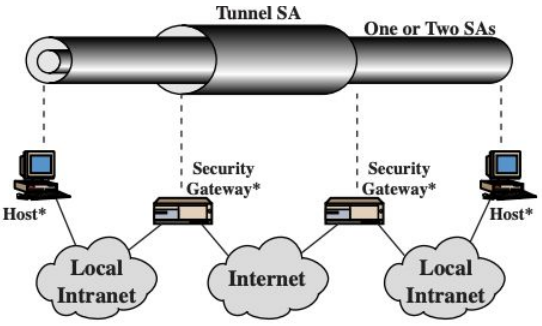


(b) Case 2

Case 3

Security is provided between gateways (routers, firewalls, etc.) and both hosts implement IPsec.

- Builds on case 2 by adding end-to-end security.
- Same combinations as for cases 1 and 2.
- The gateway-to-gateway tunnel provides either authentication, confidentiality, or both for all traffic between end systems.
- When the gateway-to-gateway tunnel is ESP, it also provides a limited form of traffic confidentiality.
- Individual hosts con implement any additional IPsec services required for given applications or given users by means of end-to-end SAs.

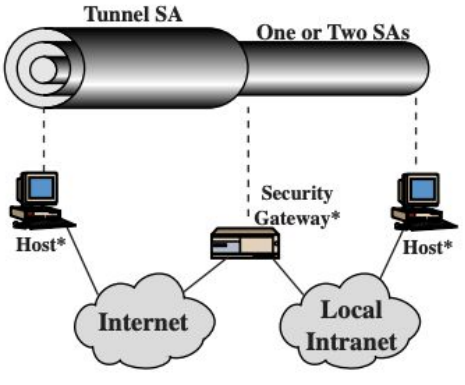


(c) Case 3

Case 4

This provides support for a remote host that uses Internet to reach an organization's firewall and then to gain access to some server or workstation behind the firewall.

- Only tunnel mode is required between remote host and firewall.
- As in case 1, one or two SAs may be used between remote host and local host.



(d) Case 4

Associations Key Management (IPSEC: IKE)

The Internet Key Exchange (IKE) protocol

IKE establishes not just keys, but Security Associations:

- the protocol format used (many),
- the cryptographic and hashing algorithm used,
- the key

IKE is very flexible: E.g., supports authentication based on a variety of pre-shared secrets (master keys). But also very complex: Many options, alternative subprotocols,

IKE evolved from a number of different protocols, including:

- ISAKMP (Internet Security Association and Key Management Protocol):
    - provides a framework and a generic negotiation protocol for establishing SAs and cryptographic keys, **but** does not prescribe any particular authentication mechanism.
- Oakley: a suite of key agreement protocols in which two parties generate a key jointly.

Roughly speaking, IKE combines packet formats of ISAKMP and exchanges of OAKLEY, which are both based on Diffie-Hellman. Bottom line: 2x Diffie-Hellman (+ extensions).

Perfect Forward Secrecy (PFS)

Diffie-Hellman protocol: if you already share keys, why bother with Diffie-Hellman?
Answer:
Perfect Forward Secrecy (PFS): A property of key-agreement protocols ensuring that a session key derived from a set of long-term keys cannot be compromised if one of the long-term keys is compromised in the future.

- The key used to protect transmission of data must not be used to derive any additional keys, and if the key used to protect transmission of data is derived from some other keying material, then that material must not be used to derive any more keys.
- In this way, compromise of a single key permits access only to data protected by that single key.

The trick to achieving PFS is to **generate a temporary session key, not derivable from information stored at the node after the session concludes**, and then forget it after the session concludes.

Example without PFS: SSL

- If server's private key is compromised, then past sessions can be decrypted.

Moreover, periodic generation of new keys and keying material complicates cryptanalysis.

IKE Protocol

IKE is a protocol suite for establishing and maintaining SAs.

Typical key establishment protocol: one phase, in which two parties use master keys to establish shared keying material.

- **Master keys**: pre-shared secret (symmetric) keys, public encryption key (used only for en/decryption), or public signature key (restricted to signing and signature verification).

IKE proceeds in two phases:

1. Two parties negotiate an SA. They agree on keying material and mechanisms (e.g., crypto-algorithms, hash functions, ...) to use in phase 2.
2. The SA of phase 1 is used to create “child SAs” to encrypt and authenticate further communications.

We just consider phase 1 here.

IKE Phase 1

Establish a secure channel so that phase 2 negotiations can occur privately.

Negotiate a “ISAKMP” SA using Diffie-Hellman (DH) exchange to then exchange:

- **Cookies:** to protect against DOS attacks.
- **Identifiers**: to name partners (typically IP addresses).
- **Authenticators**: data computed by a MAC, *h(key, data)*

All of this occurs after exchanging a key with Diffie Hellamn

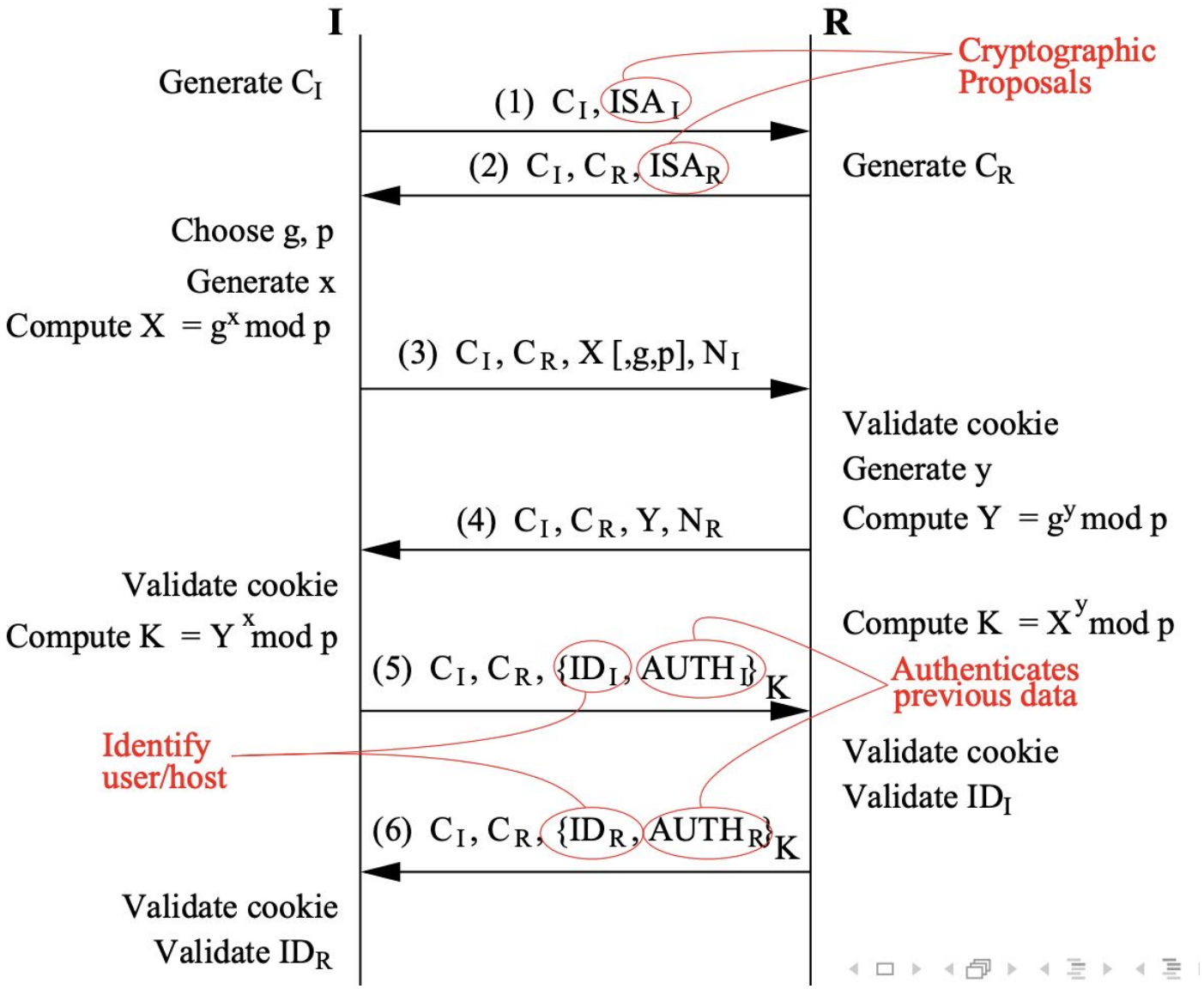
Typically (= main mode), exchange of authenticated IDs occurs after DH exchange, so that DH key material can be used to encrypt the messages in the authentication phase.

Phase 1 offers **two modes** (both resulting in the desired SA):

- Main mode:
    - 6 messages exchanged between I and R.
    - Offers identity protection and considerable flexibility in terms of the parameters and configurations that can be negotiated.
- Aggressive mode: is faster but without protection.
    - 3 messages exchanged between I and R.
    - This exchange occurs without identity protection, except when public-key encryption is used for authentication. The exchange doesn’t care that much about who is the original communicator.

Each mode has 4 variants, based on the authentication method (pre-shared key, digital signatures, and 2 public-key variants). The derivation of keying material and message contents depend on the authentication scheme used.

Main Mode

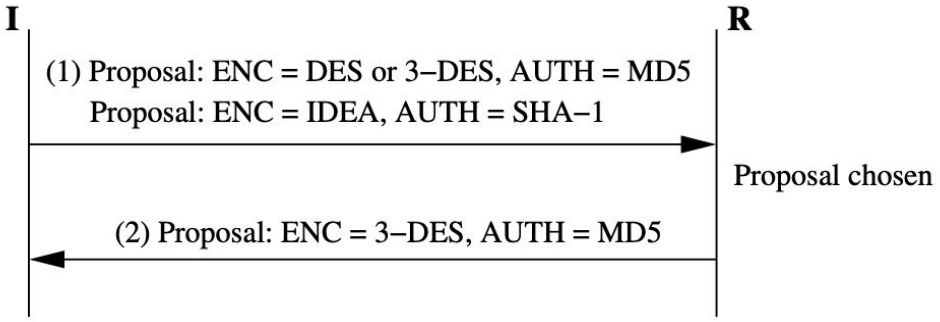


C_I: cookie from I
g: generator
p: large prime number
NI: nonce
X: half-key
ID_R: identifier of user/host
AUTH: authenticates previous data

$$(1) \quad I \rightarrow R: \quad C_I, ISA_I$$
$$(2) \quad R \rightarrow I: \quad C_I, C_R, ISA_R$$

1: ISA_I (ISAKMP SA data from I) includes a series of **proposals** and a list of algorithms for each proposal.
2: ISA_R (ISAKMP SA data from R) consists of the proposals that were accepted and the algorithm chosen for each.

Example negotiation (fails if R does not support any of I's proposals):



$$
\begin{array}{lll}
(1) & I \rightarrow R: & C_I,\ ISA_I \\
(2) & R \rightarrow I: & C_I,\ C_R,\ ISA_R \\
(3) & I \rightarrow R: & C_I,\ C_R,\ X\,[, g, p],\ N_I \\
(4) & R \rightarrow I: & C_I,\ C_R,\ Y,\ N_R \\
(5) & I \rightarrow R: & C_I,\ C_R,\ \{ID_I,\ AUTH_I\}_K \\
(6) & R \rightarrow I: & C_I,\ C_R,\ \{ID_R,\ AUTH_R\}_K
\end{array}
$$

Messages (3) and (4) contain the half-keys and nonces of I and R.
Messages (5) and (6) are encrypted with the algorithm negotiated by the ISAKMP SA of (1) and (2)

- Key K determined from Diffie Hellman Key Exchange, this is a session key.
- ID_I and ID_R are ways of identifying each user or host (e.g., an IP address, fully-qualified domain name, certificate, or email address).
- $AUTH_I$ and $AUTH_R$ is authentication data, depending on variant used, e.g., MAC or signature.

Concrete example of authentication variant using digital signatures:

$$
\begin{array}{lll}
(1) & I \rightarrow R: & C_I,\ ISA_I \\
(2) & R \rightarrow I: & C_I,\ C_R,\ ISA_R \\
(3) & I \rightarrow R: & C_I,\ C_R,\ g^x,\ N_I \\
(4) & R \rightarrow I: & C_I,\ C_R,\ g^y,\ N_R \\
(5) & I \rightarrow R: & C_I,\ C_R,\ \{ID_I,\ SIG_I\}_{SKEYID_e} \\
(6) & R \rightarrow I: & C_I,\ C_R,\ \{ID_R,\ SIG_R\}_{SKEYID_e}
\end{array}
$$

$$
\begin{array}{lll}
SKEYID & = & h(\{N_I, N_R\}, g^{xy}) \ \text{(h is keyed hash)} \\
SKEYID_d & = & h(SKEYID, \{g^{xy}, C_I, C_R, 0\}) \quad \text{deriving key} \\
SKEYID_a & = & h(SKEYID, \{SKEYID_d, g^{xy}, C_I, C_R, 1\}) \quad \text{authentication key} \\
SKEYID_e & = & h(SKEYID, \{SKEYID_a, g^{xy}, C_I, C_R, 2\}) \quad \text{encryption key} \\
HASH_I & = & h(SKEYID_a, \{g^x, g^y, C_I, C_R, ISA_I, ID_I\}) \\
HASH_R & = & h(SKEYID_a, \{g^y, g^x, C_R, C_I, ISA_R, ID_R\}) \\
SIG_I & = & \{HASH_I\}_{K_I^{-1}} \\
SIG_R & = & \{HASH_R\}_{K_R^{-1}}
\end{array}
$$

Deriving key: used to create other keys.
Auth key: used to authenticate parties.
Enc key: used to encrypt data.

IKE phase 2

Protected by the SA established in phase 1, establish new SAs that can be used to protect “real” communication (i.e., the IPsec SA).

- **Quick mode**: Establish new SAs (e.g., for ESP or AH).
- **New Group mode**: Negotiate new parameters of the SAs (e.g., change cryptographic group). This mode does not establish new SAs (and it does not replace or preempt quick mode).

You compute a new key to be used in the actual communication.

SSL

SSL: The SSL protocol provides communications privacy over the Internet. The protocol allows client/server applications to communicate in a way that prevents eavesdropping, tampering, or message forgery.

It is built to provide security on top of the application layer.

The goal is to achieve secrecy, integrity, (optionally mutual, normally we are just interested in the server authenticating to us) authentication. Works by setting up a one (or two) way authentic channel for secret communication using public-key certificates.

Original protocol (SSL 2.0) developed by Netscape (1995) to allow credit cards to be exchanged over the Internet. Current version TLS, based on SSL 3.1. Used for many applications, e.g., HTTP(S), SMTP, IMAP, …

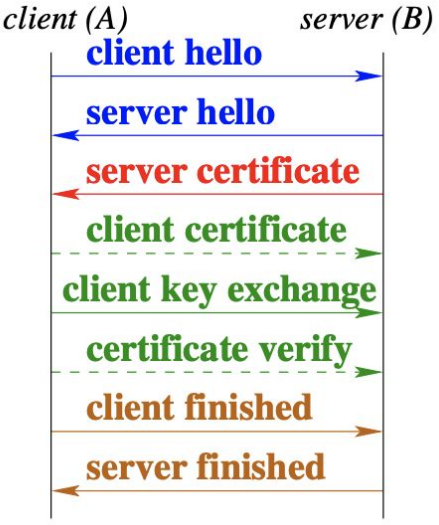
We are interested in that a server authenticates himself. Ie if we are connecting to the bank, we want to make sure we are really communicated with the bank server.

SSL consists of various subprotocols, which run on top of TCP.

- **SSL Handshake**: initiates (or reinitialized existing) connection. Establishes channel with desired channel properties.
- **SSL Record:** protocol for sending application data. Describes way in which application data is compressed, authenticated with a MAC, and resulting payload is encrypted.
- **Other protocols:** for renegotiating ciphers and error recovery

The underlying concepts are:

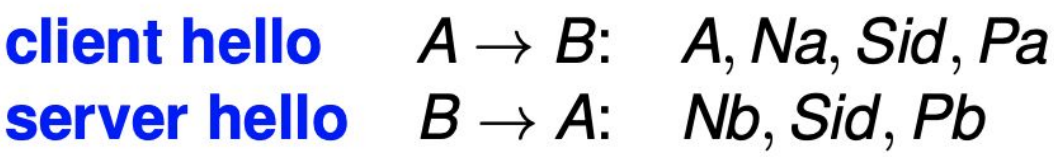
- An **SSL session** is an association between the client and server with an associated state that specifies encryption method, client and server MAC secrets, encryption keys, etc.
- An **SSL connection** is a secure stream within a session.



SSL Handshake

1. Establish security capabilities. Negotiate ciphers (DH, RSA, ...).
2. Exchange server certificate.
3. Client key exchange. Optional authentication of client using client certificate.
4. Finish, establishing session.

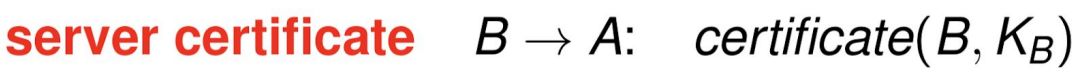
Handshake - Hello



- ***A*** identifies client. In practice, IP address used from TCP.
- ***Sid*** is session identifier.
- ***Pa*** is list of A's preferences for encryption and compression. (E.g., Diffie-Hellmann with signatures, RSA key exchange, …)
- ***Pb*** is chosen from *Pa*

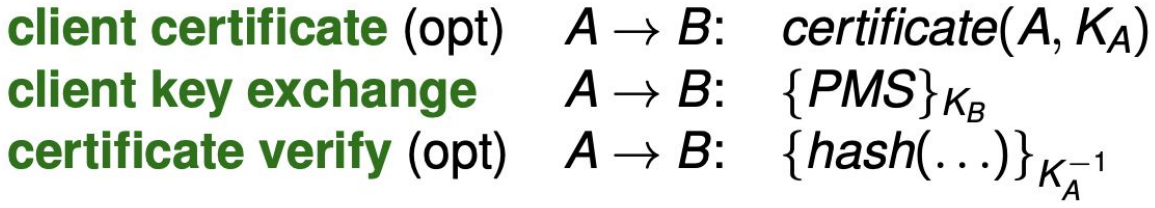
The items Pa and Pb are unprotected and authenticated later.

Handshake - Server certificate



- Certificate: X.509v3, signed by trusted 3rd party. Here details other than the name and public key are omitted.
- In what follows, we will make additional, slightly simplifying assumptions for initial key exchange (next slide).
    - Actual protocol allows additional key-exchange message, needed if negotiating, e.g., a DH key.
    - In full protocol (here omitted), server may also request a certificate from client too. Rarely used in practice as it requires clients have personal public key certificates.

Handshake - client exchange



- ***PMS*** is a pre-master secret used to compute a **master secret M,** which is in turn used to generate various keys.
  E.g., *M = PRF(PMS, Na, Nb)*, where PRF is a pseudo-random function, e.g., constructed from a MAC.

The first and third messages authenticate the client. Client sends certificate and hash(. . .), computed using all previous messages exchanged

Handshake - finish

client finished $A \rightarrow B$: $\{Finished\}_{clientK}$
server finished $B \rightarrow A$: $\{Finished\}_{serverK}$

- ***Finished*** is a hash of previous messages: *M, Na, . . .* It guarantees integrity of previous items. E.g., prevents downgrading attack on chosen ciphers or man-in-the-middle attack.
- ***clientK*** and ***serverK*** are symmetric keys for client and server encryption/decryption.
    - Each generated from Na, Nb, and M.
    - 4 other keys are also generated: 2 for MACS and 2 initialization vectors for stream cipher (for encrypting data, including MAC).
- Subsequent messages authenticated/encrypted using these keys.

The more important exchanges are:

1. **server certificate** $B \rightarrow A$: $certificate(B, K_B)$
2. **client key exchange** $A \rightarrow B$: $\{PMS\}_{K_B}$

- If A shares one-sided key with a trusted CA and has a communication channel with B then (1) promotes this to an authentic channel (these CAs are typically stored in your browser)
- In (2), a non-authenticated agent A sends keying material to B. Effect of encryption/MAC-ing with this can be understood as follows:
    - When A subsequently sends: we have sender invariant channel. When B subsequently responds: channel is receiver invariant.
    - Both yield essentially a secure channel with a pseudonym.
- Secure channel with a pseudonym cannot be promoted to a secure channel, where sender's identity is authenticated.
  Hence SSL often followed by additional client authentication.

The whole thing relies on whether you trust the certificate from a server or not.

SSL for web servers

Recall: Netscape's motivation: Support secure communication of credit card numbers and customer data with web servers.

Requires that both browser and server are SSL-enabled. Various libraries available: SSLRef, OpenSSL,

Objective is user-friendly security for the masses.

- Does a closed lock in browser actually provide this?
- What are some of the vulnerabilities in practice?
- But does the lock mean SSL? Do people click on it?
- Users tend to skip warning messages without knowing if they should be skipped or not

Possible attacks: phishing to do identity theft.

Some of the solutions include multifactor authentication of the client. Unfortunately, phishing can occur server-side. A possible solution is to do communication over multiple channels:

- SSL combined with GSM
- Server sends the user a TAN by SMS required for authentication.

In this case the attack is still possible but requires a man in the middle in both channels. Unfortunately, this requires that our server has an existing relationship with the client's phone.

Lecture 9 Privacy

Privacy: *what you choose what you let other people know.* With privacy you want to control who can see your data therefore it's more than confidentiality. Controlling the sharing of information, re-distributing of information and the length of time for which people hold your data.
Anonymity: *a condition in which your true identity is not known. The confidentiality of your identity.*

Privacy and anonymity on public networks

Internet is designed as a public network.

- Machines on your LAN may see your traffic, network routers see all traffic that passes through them.
- Email is not a letter but rather a post card! (Everyone can read it along the way.)

Routing information is public.

- IP packet headers identify source and destination.
- Even a passive observer can easily figure out who is talking to whom.

Encryption does not hide identities.

- Encryption hides payload, but not routing information.
- Even IP-level encryption (tunnel-mode IPsec/ESP) reveals IP addresses of IPsec gateways.

Why is anonymity difficult?

- Packet headers identify recipients
- Packet routes can be tracked (traffic analysis)
- Payload, even when encrypted, is visible.

Applications of Privacy and Anonymity

Privacy:

- You want to hide online transactions, web browsing from intrusive governments, marketers and archivists.

Untraceable electronic mail:

- Confidential business negotiations
- Political intelligence
- Corporate whistleblowers

Digital cash:

- Electronic currency with properties of paper money which is unlinked to the buyer's identity

Anonymous electronic voting
Censorship-resistant publishing

Anonymity given that actions can be observed, you are only anonymous within a group if your actions (sending, receiving and communication) cannot be distinguished from anyone else in a group.

This group is called the **anonymity set**, the larger the better. *Being anonymous by yourself is difficult*

- This shows that you need anonymity by yourself, you need other people.

Anonymity: is the state of being not identifiable within a set of subjects. Namely, hide your activities among others' similar activities.

Unlinkability of action and identity. For example, sender and his email are no more related after observing communication than they were before.

Unobservability (hard to achieve). Observer cannot even tell whether a certain action took place or not.

Attack on Anonymity

Assuming that everything can be attacked:

Passive traffic analysis:

- Infer form network traffic who is talking to whom.
- To hide your traffic, must carry other people's traffic

Active traffic analysis:

- Inject packets or put a timing signature on packet flow

Compromise of network nodes (routers):

- It is not obvious which nodes have been compromised
    - Attacker may be passively logging traffic.
- Better not to trust any individual node
    - Assume that some fraction of nodes is good, don't know which

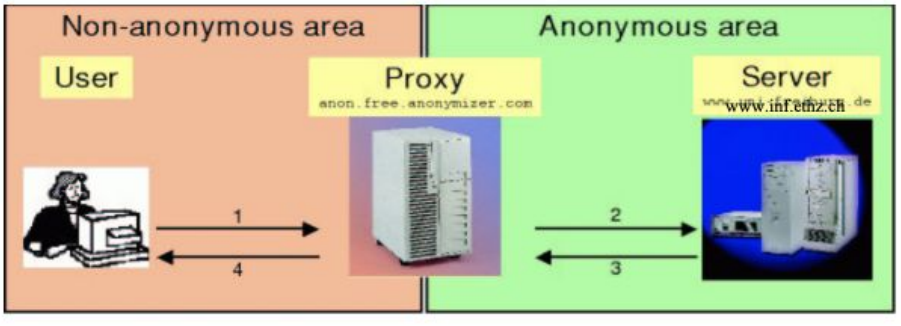
Unobservability/Anonymity

We must anonymize the sender and/or the receiver. Provide confidentiality of principals' identities.

We can use **pseudonyms** as a lightweight mechanism. Naming resolving might be quite problematic, imagine for instance Batman. The point of this is to put some difficulty in the mapping between who is doing something and who that person is.

Sender/recipient anonymity: proxies

Sender: To remain anonymous as a sender, you use a proxy, you relay the information from the user to the server.



Weaknesses:

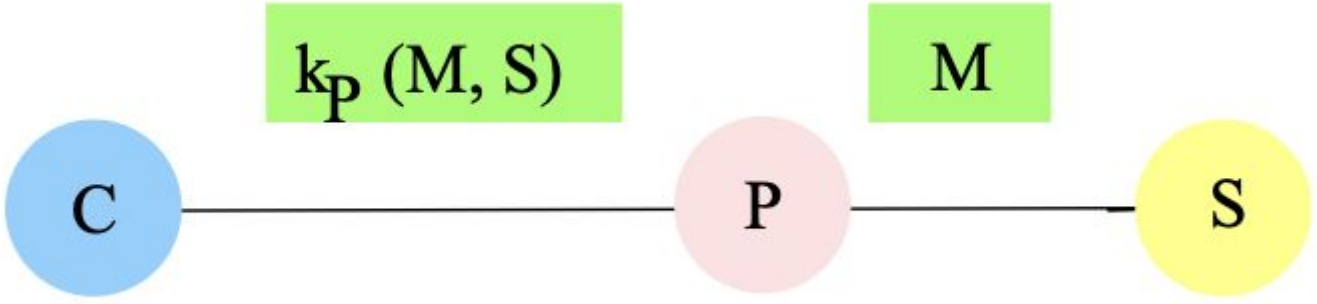
- The proxy knows everything about the packets
- Somebody looking in the proxy and looking what gets in and out can then correlate the different messages.

Solution: having a cascading proxy chain or a mixed network.

Receiver: broadcasting or multicasting (broadcast in an anonymity group), everyone received the message but only one can decrypt it.

Generalization: cascaded proxies with encryption

Here the client shares some kind of key with the proxy.



This can be generalised to a cascade of proxies which allows each proxy to add a layer of security. In this case each proxy only knows the previous and the next point thus compromising one of them is not enough. However, we can still look at every single node in the network and put the pieces together.

Mix Networks

Developed by David Chaum created untraceable electronic mail, return addresses and digital pseudonyms. They are used to send a message to someone and remain anonymous. The sender does know the receiver but the receiver does not know who the sender is.

His idea he had was to use public key and a trusted proxy/re-mailer (Mix).

- Public keys used as persistent pseudonyms.
- Untrusted communication medium: designed to work in environment with an active attacker who
    1. can learn origin, destination(s), and representation of all messages in the communication system;
    2. can inject, remove, or modify messages;
    3. however, **cannot determine anything about the correspondences between a set of encrypted items and the corresponding set of unencrypted items, or create forgeries.**

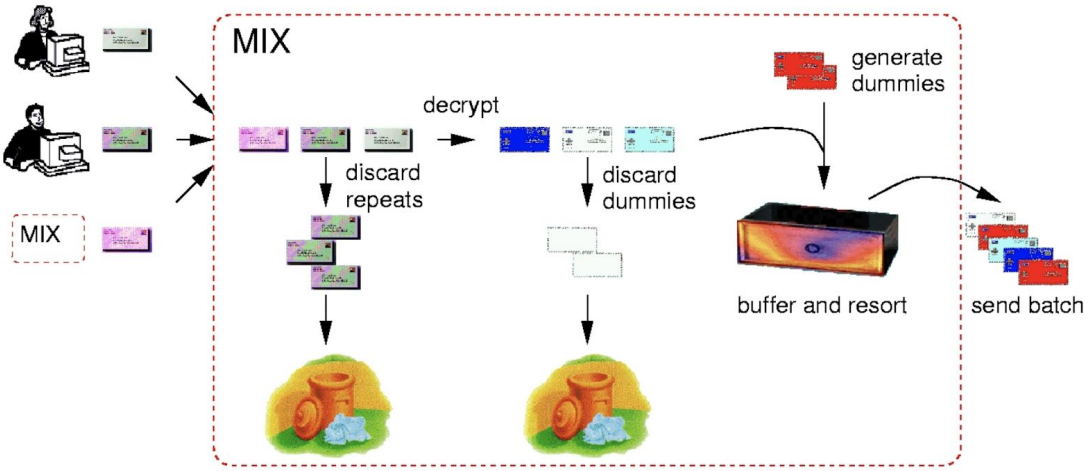
A **mix** is a server that processes items.
To send message M to agent at address B:

$$\{r_1, \{r_0, M\}_{pk(B)}, B\}_{pk(mix)} \longrightarrow \{r_0, M\}_{pk(B)}, B$$

The mix generates output and sends it to component B, only B can decrypt it as we encrypt it with B's public key so that only the mix can decrypt it. r_i are padding strings of random bits. The mix can also perform additional operations to foil traffic analysis.

Foiling traffic analysis: four requirements

1. Agents/mixes work with uniformly sized items.
   - I.e., messages split (or padded) into fixed size blocks.
2. Order of arrival hidden by outputting items in batches.
   - Can use fix ordering (e.g., lexicographic) or random ordering.
3. Repeated information must be blocked.
   - So Mixes must filter duplicates, cross-checking across batches.
   - Or strings r include, e.g., a time stamp.
4. Sufficient traffic from a large anonymity set is required.
   - Few clients sending entails weak anonymity.
   - Solution involves clients regularly sending (and receiving) dummy messages.



If desired, a Mix can return a receipt Y for messages received

$$Y = \{c, \{r_1, \{r_0, M\}_{pk(B)}, B\}_{pk(mix)}\}_{priv(mix)}$$

where c is some large, known, constant (e.g., string of zeros).

Participant can later prove he sent message by supplying

$$X = \{r_0, M\}_{pk(B)}, B$$

as well as retained string r_1.

Proof of receipt:

$$\{Y\}_{pk(mix)} = c, \{r_1, X\}_{pk(mix)}.$$

A single mix has the same weakness as a single proxy, if you compromise it, you know everything. That is why this weakness is reduced by forming **Mix networks (or cascades)**. Messages are sent through a sequence or

network of mixes, each mix in a different administrative domain. Although some mixes may be controlled by an attacker, a good **single** mix guarantees anonymity.

Sending a message through n number of mixes

Sender prepared item for every Mix in cascade (n mixes):

$$\{r_n, \{r_{n-1}, \ldots, \{r_1, \{r_0, M\}_{pk(B)}, B\}_{pk(mix_1)} \ldots, B_{n-2}\}_{pk(mix_{n-1})}, B_{n-1}\}_{pk(mix_n)}$$

Each Mix peels off (decrypts) the outermost “layer”, forwarding result.

$$\{r_{n-1}, \ldots, \{r_1, \{r_0, M\}_{pk(B)}, B\}_{pk(mix_1)} \ldots, B_{n-2}\}_{pk(mix_{n-1})}$$

Final Mix processes same data as in single-mix case.

$$\{r_1, \{r_0, M\}_{pk(B)}, B\}_{pk(mix_1)}$$

Untraceable return address

In the communication, we do not know who sent us this information, so how can we respond? To do so we make use of the untraceable return address which is inside the message.

Single Mix Case

So that the mix can respond to an anonymous sender **x** with a return message **M**.

1. The sender includes “return address”:

$$\{r_1, A_x\}_{pk(mix_1)}, pk(x)$$

   - r_1 is a random string that can also be used as a shared key.
   - pk(x) is a fresh public key, created for this purpose.
   - A_x is x’s actuall address.

1. Receiver sends to the “response” mix:

$$\{r_1, A_x\}_{pk(mix_1)}, \{r_0, M\}_{pk(x)}$$

2. Mix transforms this to:

$$A_x, \{\{r_0, M\}_{pk(x)}\}_{r_1}$$

3. Second part is sent to A_x.

Note that in this case r is used as as symmetric key.

- The encryption with r_1 masks input/output correlation.

- Only original sender can decrypt as he created both pk(x) and r_1

In the general case (with many mixes)

- Generalize return addresses of single Mix case

$$\{r_1, \{r_2, \ldots \{r_n, A_x\}_{pk(mix_n)} \ldots\}_{pk(mix_2)}\}_{pk(mix_1)}, pk(x)$$

- Recipient sends following response to 1st (return) Mix

$$\{r_1, \{r_2, \ldots \{r_n, A_x\}_{pk(mix_n)} \ldots\}_{pk(mix_2)}\}_{pk(mix_1)}, \{r_0, M\}_{pk(x)}$$

- Result of 1st Mix is

$$\{r_2, \ldots \{r_n, A_x\}_{pk(mix_n)} \ldots\}_{pk(mix_2)}, \{\{r_0, M\}_{pk(x)}\}_{r_1}$$

- Final result after n-1 mixes is

$$A_x, \{\{\{\{r_0, M\}_{pk(x)}\}_{r_1}\}_{r_2} \ldots\}_{r_n}$$

Attacks on Mix Network

Tracing messages: this can be prevented by sending dummy messages.
Replay attack: can be prevented by a replay filter not allowing multiple similar requests at the same time.
N-1 attack: what happens if an attacker knows (ie has sent himself) n-1 of the n messages input to a mix?

Advantages & Dadvantages

+ **Very high degree of anonymity.**
  - No correlation between Mix input and output.
  - Only some nonzero fraction of mixes need be honest.
  - With enough dummy traffic, the anonymity set is entire network.
+ **Cryptography used in a novel way**
  - With anonymous response, sender is anonymous even to receiver.
  - Can also be used for anonymous “certified mail” where receipt is signed by receiver and every Mix along the path.

- **Public-key encryption and decryption at each mix are computationally expensive**.
- Dummy overhead (but in a network with substantial communication, dummies can be reduced or even eliminated).
- **High latency:** OK for email but not for anonymous web browsing.

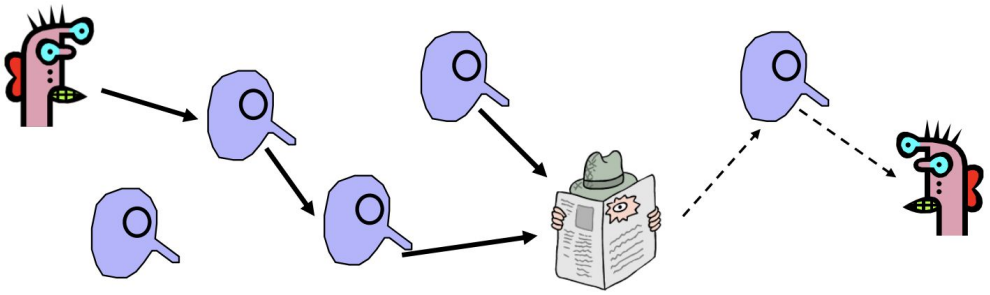
Challenge: low-latency anonymity network.

- Use public-key cryptography to establish a “circuit” with pairwise symmetric keys between hops on the circuit.
- Then use symmetric decryption and re-encryption to move data messages along the established circuits.
- Each node behaves like a mix; anonymity is preserved even if some nodes are compromised.

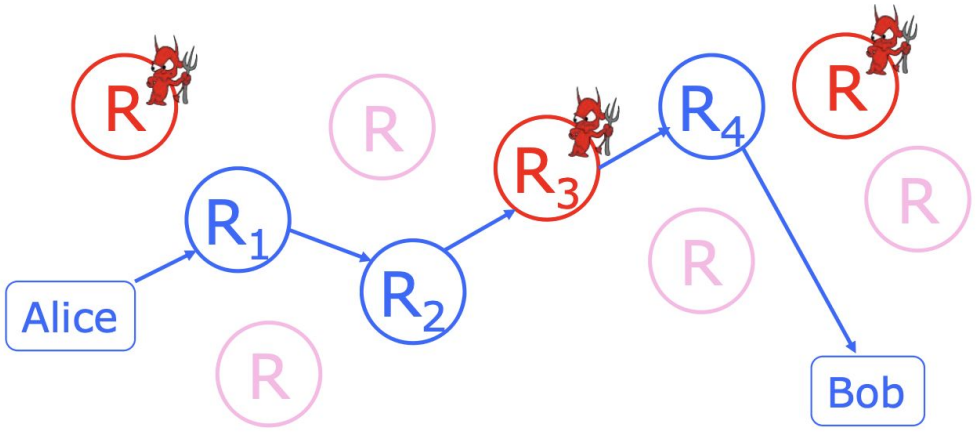
Onion Routing

Hide message source by routing it randomly.

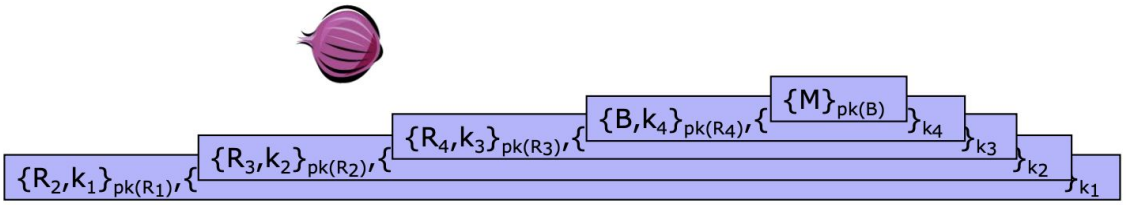
A path is established using PKE but messages are then sent Symmetric encryption.



Routers don’t know for sure if the apparent source of a message is the true sender or another router.

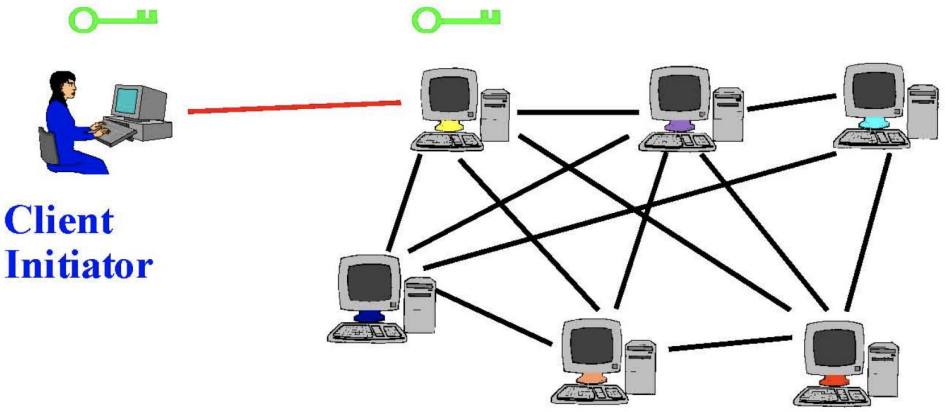


Sender chooses a random number of routers, some routers are honest and some are controlled by an attacker. The sender can control the length of the path.

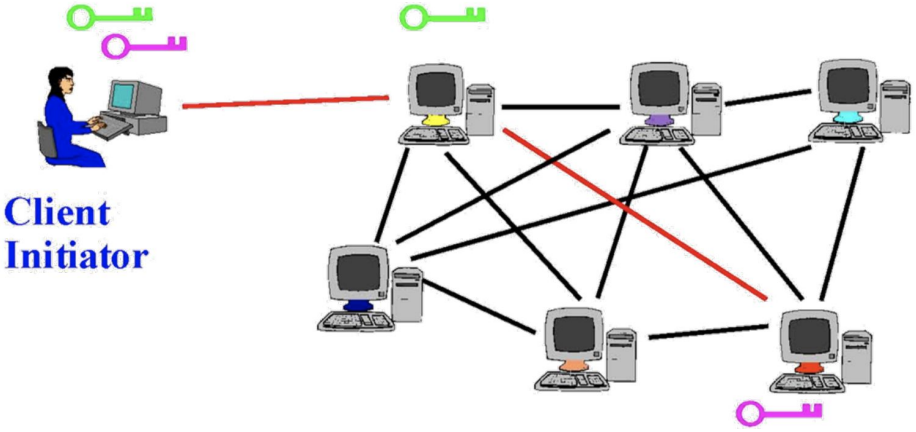


You select the nodes you want to use in the network and then use the public key of the router to encrypt with their public key. In this way each router learns only the identity of the next user, as it must know where to sen it.

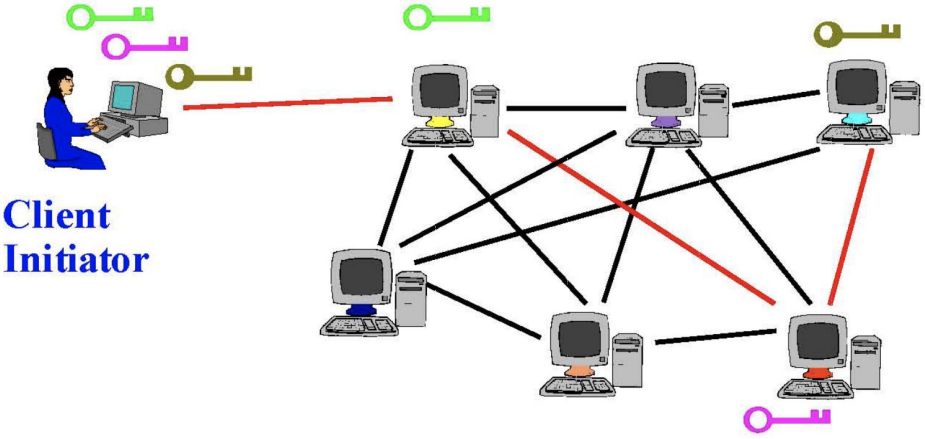
1. Client establishes a symmetric session key and circuit with Onion Router no1.



2. Client extends the circuit by establishing a symmetric session key with Onion Router no. 1 (tunnel through Onion Router no. 1)



3. Client extends the circuit by establishing a symmetric session key with Onion Router no. 3. (tunnel through Onion Router no. 1 and no. 2)



4. Client applications connect and communicate over the established Tor circuit.
    a. Datagrams are decrypted and re-encrypted at each link.

Tor Management Issues

Many applications can share one circuit.

- Multiple TCP streams over one anonymous connection.

Tor router doesn't need root privileges.

- Encourages people to set up their own routers
- More participants = better anonymity for everyone.

Directory servers.

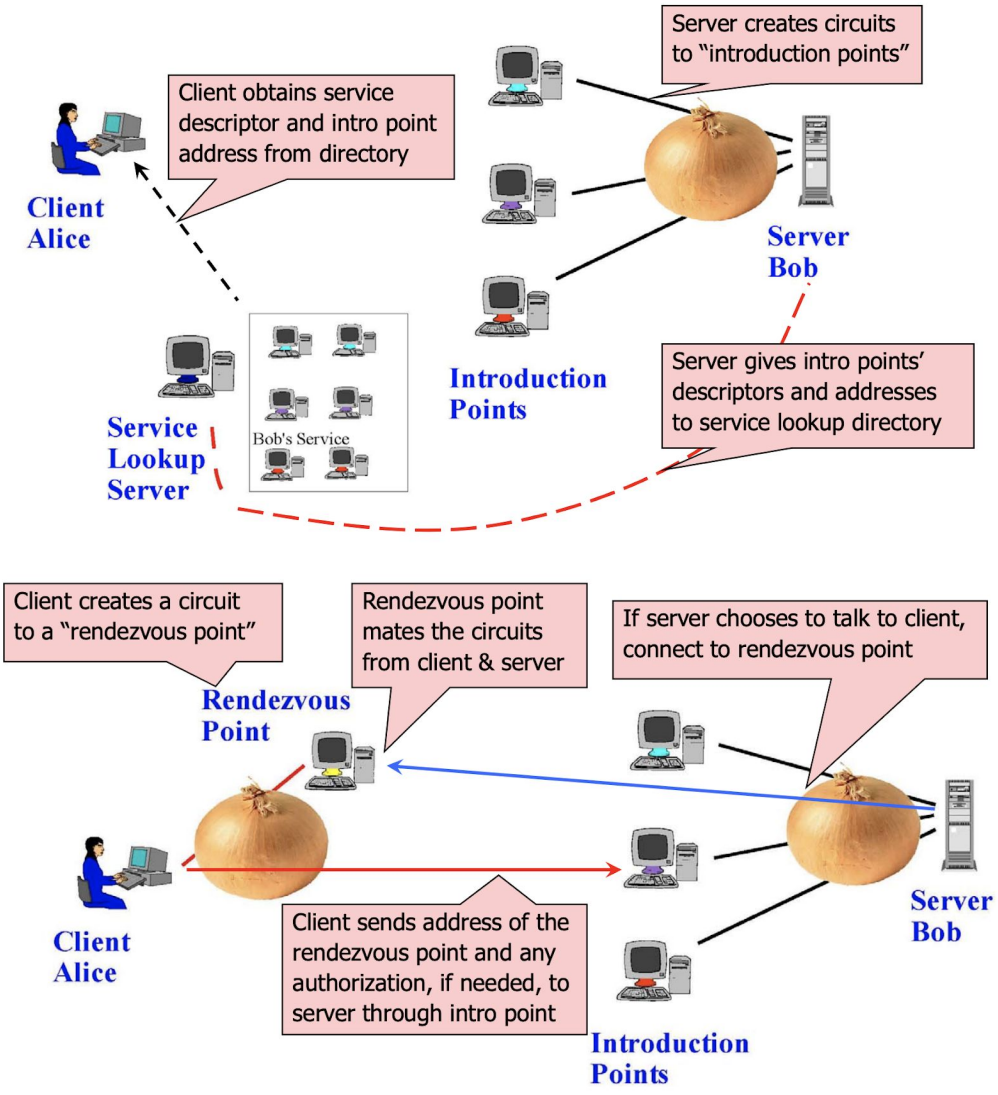
- Maintain lists of active onion routers, their locations, current public keys, etc.
- Control how new routers join the network.
    - Sybil attack: attacker creates a large number of routers and is perhaps able to identify someone.
- Directory servers' keys ship with Tor code.

Location Hidden Servers

Goal: deploy a server on the Internet that anyone can connect to without knowing where it is or who runs it.

- Accessible from anywhere
- Resistant to censorship.
- Can survive a full DoS attack
- Resistant to physical attack (as you cannot find the server).

These are typically used for the dark web, Silk Road, Darket…



Dining Cryptographers

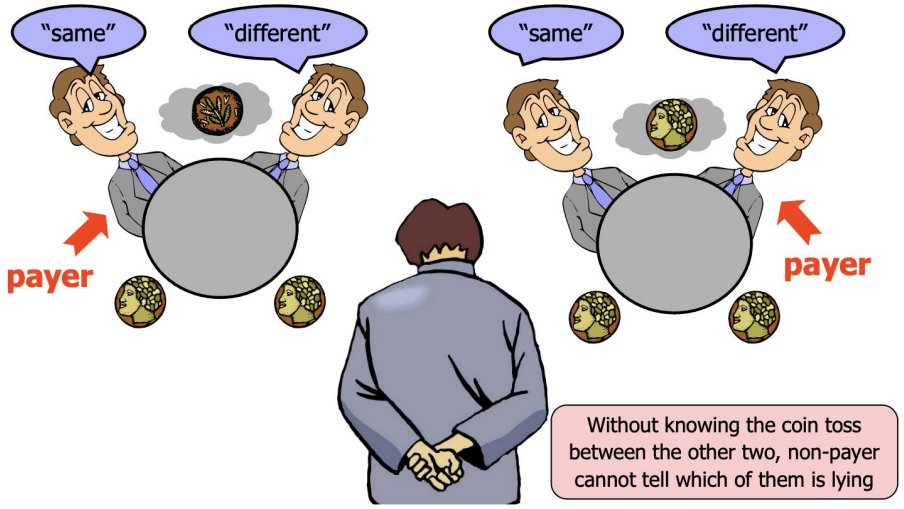
Looks at a problem in which someone can let out information to the public in a perfectly untraceable manner. This is difficult in practical as to send 1 bit we would require N number of bits (where N is the group size).

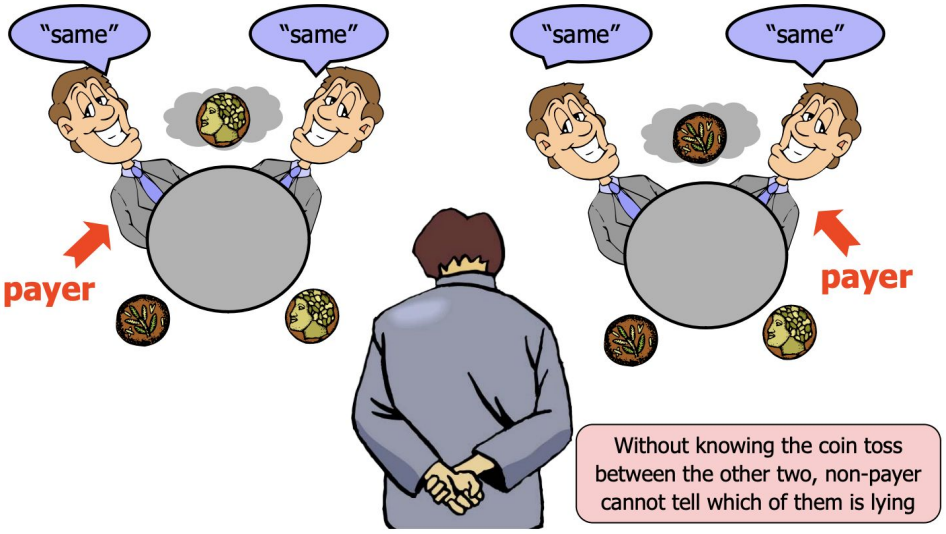
Three-person DC

Three cryptographers are having dinner.

Either **NSA is paying for the dinner, or one of them is paying, but wishes to remain anonymous**

1. Each diner flips a coin and shows it to his left neighbor
    a. Every diner will see two coins: his own and his right neighbor's.
2. Each diner announces whether the two coins are the same.
    a. If he is the payer, he lies (says the opposite)
3. If there is an odd number of "same" then the NSA is paying else there is a payee on the table.
    a. But a non-payer cannot tell which of the other two is paying (ie lying)!





The 3 person DC in Full

So, summarizing, the Three-person DC Protocol is as follows:

1. Each of the three cryptographers C_1, C_2 and C_3 flips an unbiased coin keeping the result b_i secret ($i \in \{1, 2, 3\}$).
2. Each cryptographer whispers the result b_i in the ear of the person to their immediate left.
3. Each cryptographer computes $d_i = b_i \oplus b_{i-1}$ where
   - b_i is the cryptographer's own coin flip and
   - b_{i-1} is the coin flip of the person on the right.

   Note that $d_0 = d_3$ and $b_0 = b_3$.
4. A cryptographer that did not pay for the meal announces her own d_i.
   A cryptographer that did pay for the meal lies by announcing the negation of d_i, i.e. $d_i \oplus 1$.

Note that the protocol would work as well if they all sent the bit to their right neighbour.

This idea generalizes to any group of size N.

For each bit of the message, every user generates 1 random bit and sends it to 1 neighbor.

- Every user learns 2 bits (his own and his neighbor's).

Each user announces own bit XOR neighbor's bit.

Sender announces own bit XOR neighbor's bit XOR message bit.

XOR of all announcements = message bit.

- Every randomly generated bit occurs in this sum twice (and is canceled by XOR), message bit occurs once.

Privacy: requirements, policy, and mechanisms for data protection

Anonymity: unobservability of our actions when they occur.
Data Protection: ensuring that our collected data is not distributed and used in undesired ways.

As unobservability/anonymity is more-or-less understood, let us now focus on data protection: controlling the dissemination and usage of sensitive personal data.

Confidentiality in that point is not broken as you are giving them access to your data.

A privacy policy specifies how data may be used, under which conditions, and what obligations this entails.

Privacy requirements:

- Access control requirements. *Only X may access Y.*
- Actions required before the access. Gathering owner consent.
- Actions that must be performed within a certain time period. I*nforming data owner whenever the data is used.*
- Restrictions on the further distribution of the data. *Own use only.*
- Restriction of purposes for which data may be used. *Statistical purposes only.*
- Limitations on retention time. *Delete after 7 days.*
- Mandatory use of protection mechanisms. *Encrypt backups.*
- Duties of keeping the data up-to-date. *Update every 30 days.*

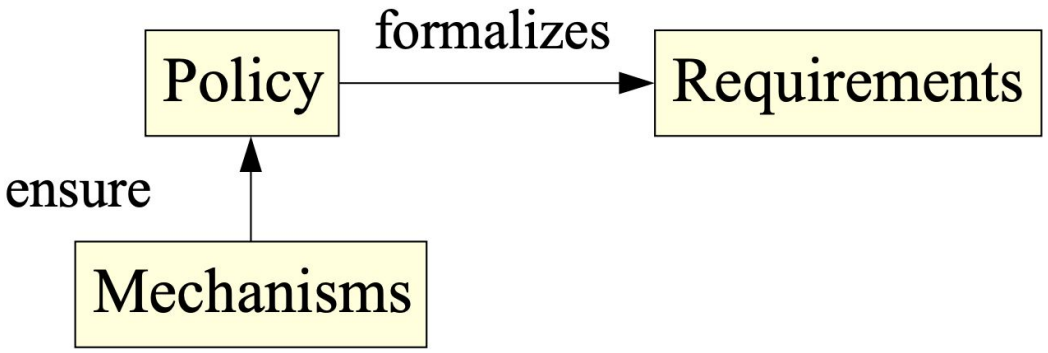
- Current mechanisms based on company/legal processes.
- Example: Online Privacy Alliance.
  - 50+ large ("global players") US companies.
  - Set of guidelines addressing collection, use and disclosure of individually identifiable information and data, as well as special measure to protect privacy of children on-line.
  - Self-enforcing, with third-party monitoring and audit.
- While a good first step, such best efforts have limited effectiveness.
  - Analogous to self restraint in conventional (CIA) security.
  - Policies prone to misunderstanding, misuse, and abuse.

We can enforce privacy reqiurements by using a policy. A privacy policy specified **how** data may be used, under which **conditions** and what **obligations** this entails.

Examples of privacy requirements:

- Access control requirements. Only X may access Y.
- Actions required before the access. Gathering owner consent.
- Actions that must be performed within a certain time period. Informing data owner whenever the data is used.
- Restrictions on the further distribution of the data. Own use only.
- Restriction of purposes for which data may be used. Statistical purposes only.
- Limitations on retention time. Delete after 7 days.
- Mandatory use of protection mechanisms. Encrypt backups.
- Duties of keeping the data up-to-date. Update every 30 days

But how are these requirements formalized and enforced?



There exist **Privacy Policy Languages** like P3P whcih allows web site to communciate their privacy policies in co mputer readable formt (XML).

This enables tools (in browsers or stand-alone) to be developed that:

- summarize privacy policies,
- compare policies with user preferences, and
- advise and alert users

Other privacy languages:

- Security Assertion Markup Language (SAML).
- Microsoft security metadata to SOAP.
- IBM’s Enterprise Privacy Authorization Language (EPAL).
- Permission-based attributes in Liberty Alliance

Current mechanisms based on company/legal processes.