January 11, 2016

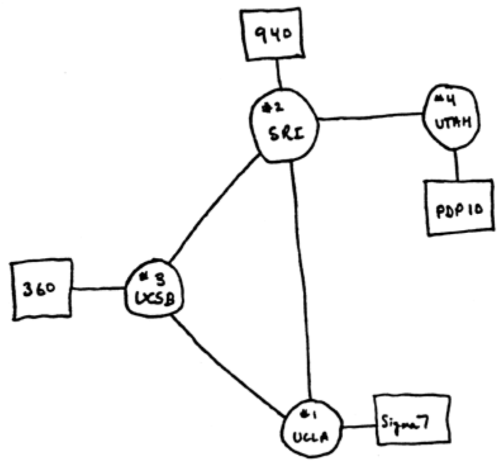
Network Security

Computer Security 2 (GA02)

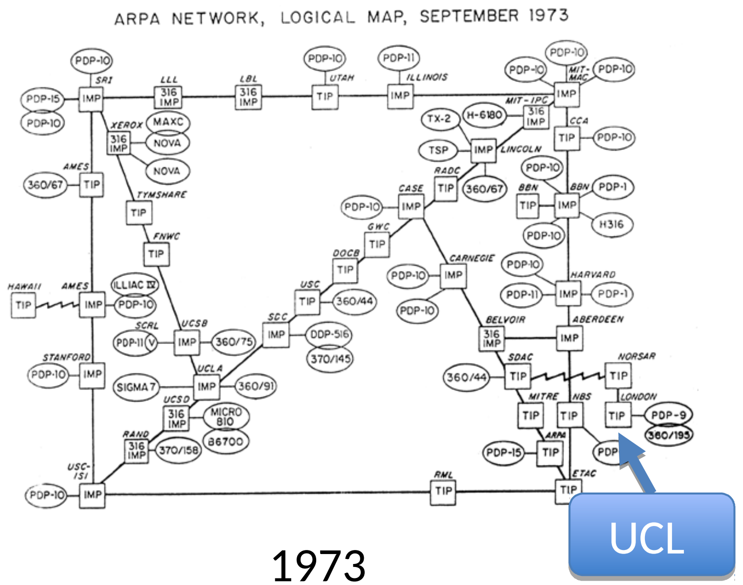
Emiliano De Cristofaro, Gianluca Stringhini

Thanks to Giovanni Vigna
for some of the materials

The Internet Was not Designed for Security



1969



1973

Standards were developed - built to make things work, not to keep things secure – built by academics, who trust each other (think of the peer review system)

- Things are getting better – encryption and authentication (SSL, IPSec, DNSSEC)
- Different ways in which standards are implemented leads to security issues
- Still important legacy and compatibility is required
- Lots of heterogeneity: vendors, operating systems, software versions

Postel's Law: be conservative in what you send, but liberal in what you accept

Bad Network Protocol Design

Common source of problems in network protocols

- **No authentication** – hosts believe to the packets they receive → **spoofing, hijacking**
- **No timestamps** – attackers can record legitimate packets and send them again → **replay attacks**
- **Insufficient checks** – attackers can send additional information that hosts never requested → **poisoning**
- **Poor implementations** – developers don't handle corner cases correctly → **memory corruption, weak randomness**

Fixing some of these problems requires re-deploying millions of servers, and is therefore difficult to achieve → **legacy requirements**

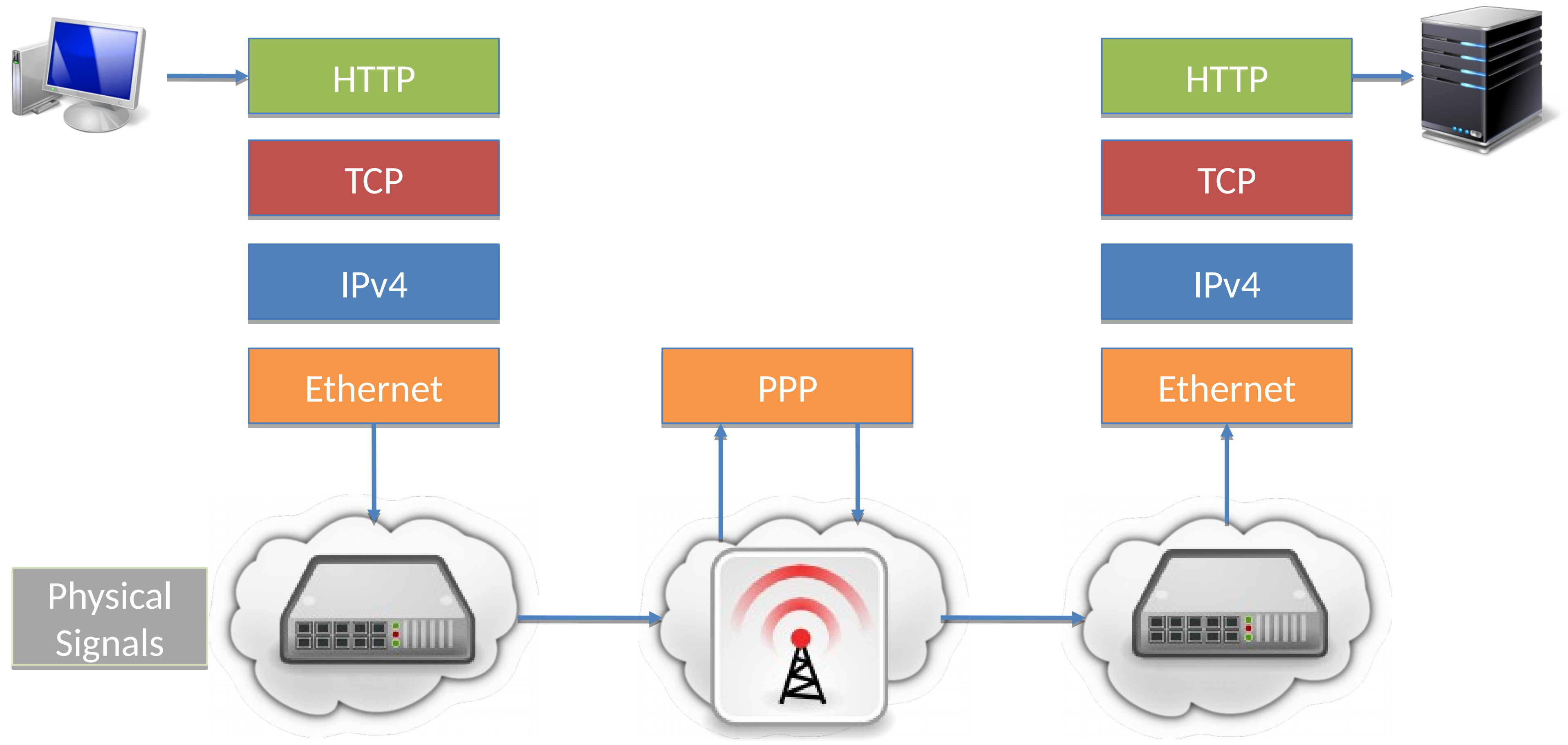
The TCP/IP Stack of Protocols

Application Layer	Deals with the protocol logic implemented by higher-level applications: **HTTP**, **FTP**, **DNS**
Transport Layer	Deals with data channel that applications uses for data exchange: **TCP**, **UDP**
Network Layer	Deals with data transmitted between different networks: **IPv4**, **IPv6**
Data-Link Layer	Deals with data transmitted between hosts connected to the same link: **Ethernet**
Physical Layer	How signals are physically transmitted

- Each layer is designed to be oblivious to the others
- Developers only have to worry about their application
- Each layer has security problems of its own

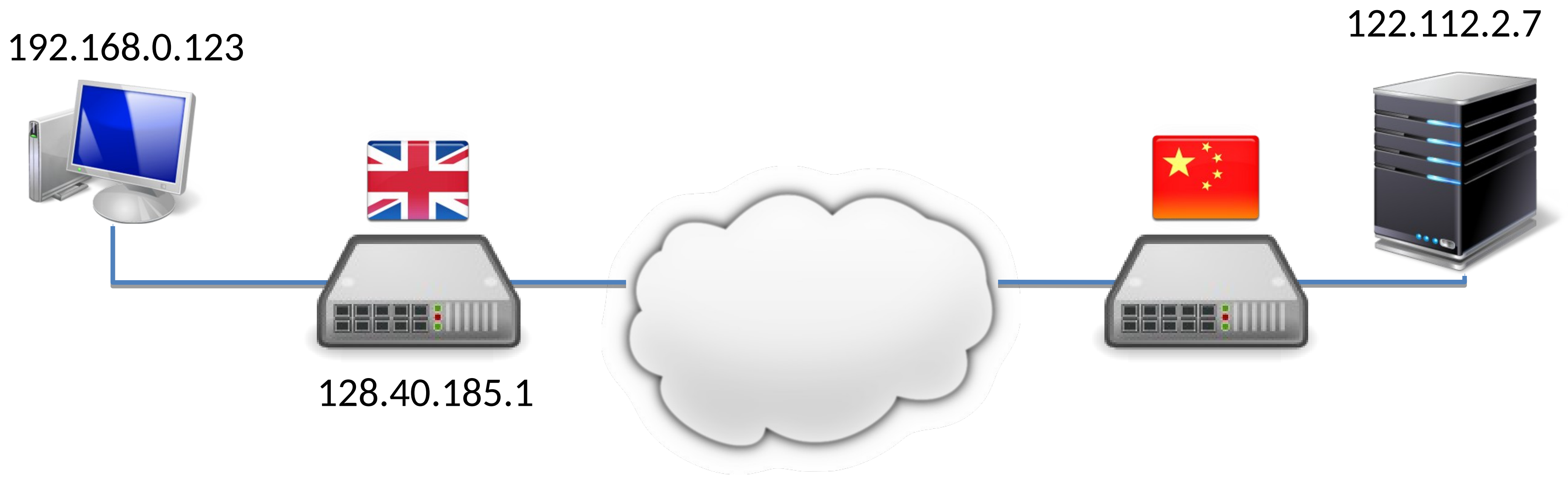
We can write secure software, but if the underlying network protocols are attackable the security of our systems is in jeopardy

Sample TCP/IP Connection



IP Addresses

Allow computers from anywhere in the world to communicate

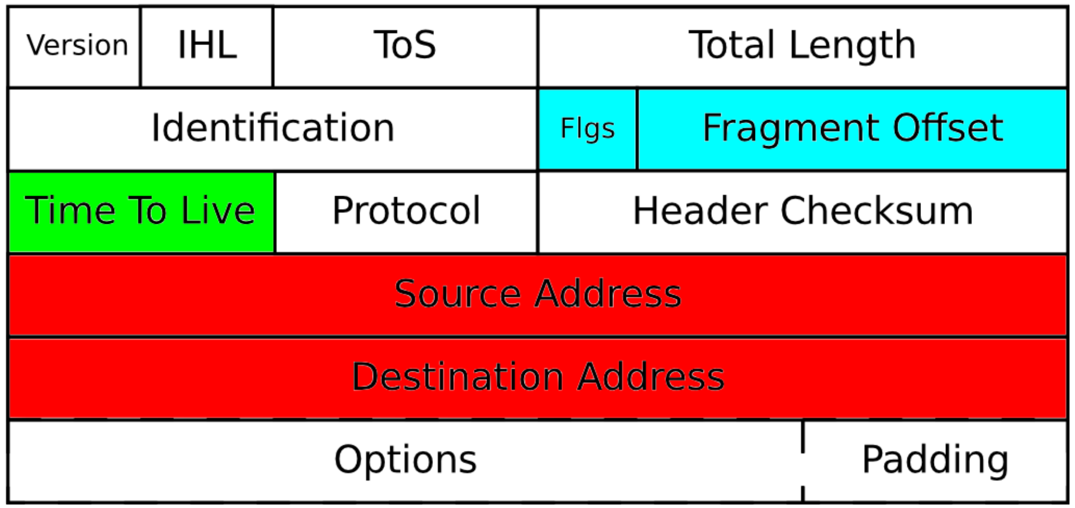


- IPv4 addresses are 32 bit long (2^{32} addresses – IPv6 adoption is needed!)
- Represented in dotted decimal notation (192.168.0.123)
- Loopback interface is local only: 127.0.0.1
- Broadcast address: all bits set to 1 (255.255.255.255) selective broadcasts to specific networks is possible too
- Private (reserved) addresses – used for your local network
  - 10.0.0.0 – 10.255.255.255
  - 172.16.0.0 – 172.31.255.255
  - 192.168.0.0 – 192.168.255.255
- Gateways have a public address and perform Network Address Translation (NAT) from private addresses

- IPv4 addresses are 32 bit long (2^{32} = 4B addresses – IPv6 adoption is needed!)
- Represented in dotted decimal notation (192.168.0.123)
- Loopback interface is local only: 127.0.0.1
- Broadcast address: all bits set to 1 (255.255.255.255) selective broadcasts to specific networks is possible too
- Private (reserved) addresses – used for your local network
  - 10.0.0.0 – 10.255.255.255
  - 172.16.0.0 – 172.31.255.255
  - 192.168.0.0 – 192.168.255.255
- Gateways have a public address and perform Network Address Translation (NAT) from private addresses

IP Datagram

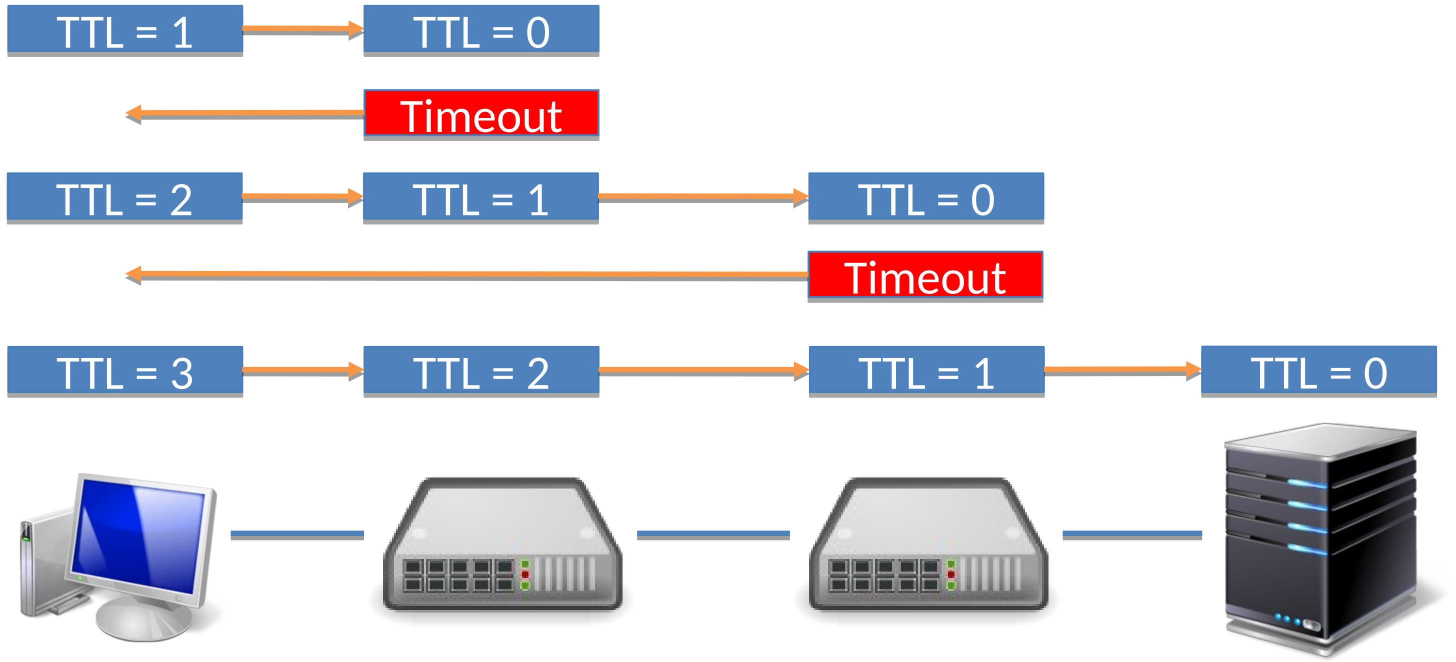
Used to encapsulate data from higher layers – best effort, no reliability guaranteed
A few fields in the IP Datagram can be misused or have security implications



- Time To Live (TTL): Decreased at each router, ensures that datagrams don't wander forever
- Flags + Offset: Maximum size of IP datagram is 65,535 bytes – datagrams can be fragmented if the lower transmission layers can handle only packets of a certain size, these fields are used to reassemble them on the receiving end
- Source, Destination address: specify where the packet comes from and has to go, no authentication provided

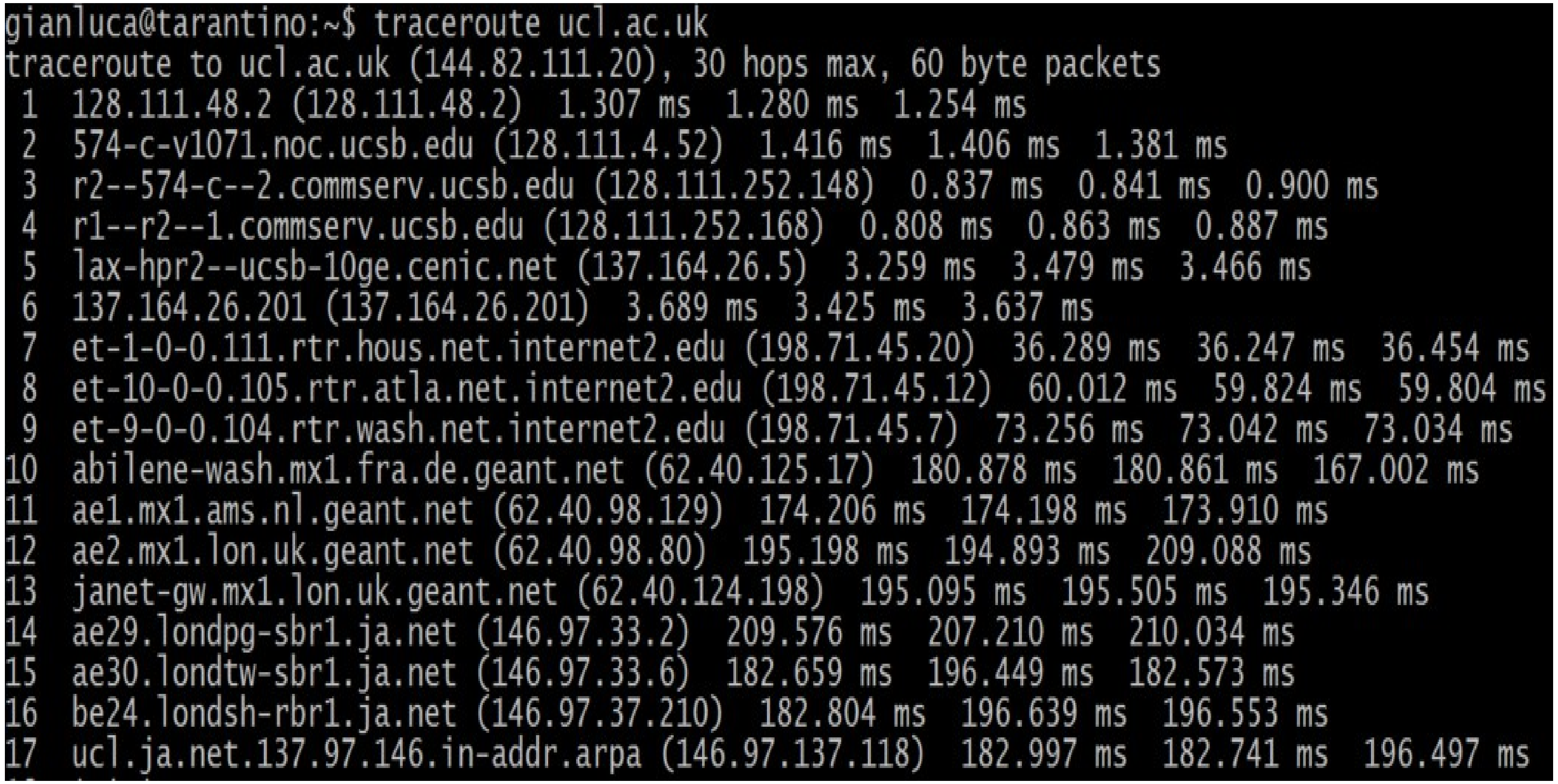
Traceroute

The Time To Live (TTL) field can be used to enumerate the routers between two hosts



- Routers decrement the TTL by one every time they route a datagram
- If TTL reaches 0, the datagram is discarded and a “ICMP Time Exceeded” packet is sent to the sender
- A host can enumerate the routers on the path to a server by sending datagrams with increasing TTL

Traceroute: Example



```
gianluca@tarantino:~$ traceroute ucl.ac.uk
traceroute to ucl.ac.uk (144.82.111.20), 30 hops max, 60 byte packets
 1  128.111.48.2 (128.111.48.2)  1.307 ms  1.280 ms  1.254 ms
 2  574-c-v1071.noc.ucsb.edu (128.111.4.52)  1.416 ms  1.406 ms  1.381 ms
 3  r2--574-c--2.commserv.ucsb.edu (128.111.252.148)  0.837 ms  0.841 ms  0.900 ms
 4  r1--r2--1.commserv.ucsb.edu (128.111.252.168)  0.808 ms  0.863 ms  0.887 ms
 5  lax-hpr2--ucsb-10ge.cenic.net (137.164.26.5)  3.259 ms  3.479 ms  3.466 ms
 6  137.164.26.201 (137.164.26.201)  3.689 ms  3.425 ms  3.637 ms
 7  et-1-0-0.111.rtr.hous.net.internet2.edu (198.71.45.20)  36.289 ms  36.247 ms  36.454 ms
 8  et-10-0-0.105.rtr.atla.net.internet2.edu (198.71.45.12)  60.012 ms  59.824 ms  59.804 ms
 9  et-9-0-0.104.rtr.wash.net.internet2.edu (198.71.45.7)  73.256 ms  73.042 ms  73.034 ms
10  abilene-wash.mx1.fra.de.geant.net (62.40.125.17)  180.878 ms  180.861 ms  167.002 ms
11  ae1.mx1.ams.nl.geant.net (62.40.98.129)  174.206 ms  174.198 ms  173.910 ms
12  ae2.mx1.lon.uk.geant.net (62.40.98.80)  195.198 ms  194.893 ms  209.088 ms
13  janet-gw.mx1.lon.uk.geant.net (62.40.124.198)  195.095 ms  195.505 ms  195.346 ms
14  ae29.londpg-sbr1.ja.net (146.97.33.2)  209.576 ms  207.210 ms  210.034 ms
15  ae30.londtw-sbr1.ja.net (146.97.33.6)  182.659 ms  196.449 ms  182.573 ms
16  be24.londsh-rbr1.ja.net (146.97.37.210)  182.804 ms  196.639 ms  196.553 ms
17  ucl.ja.net.137.97.146.in-addr.arpa (146.97.137.118)  182.997 ms  182.741 ms  196.497 ms
```

January 18, 2016

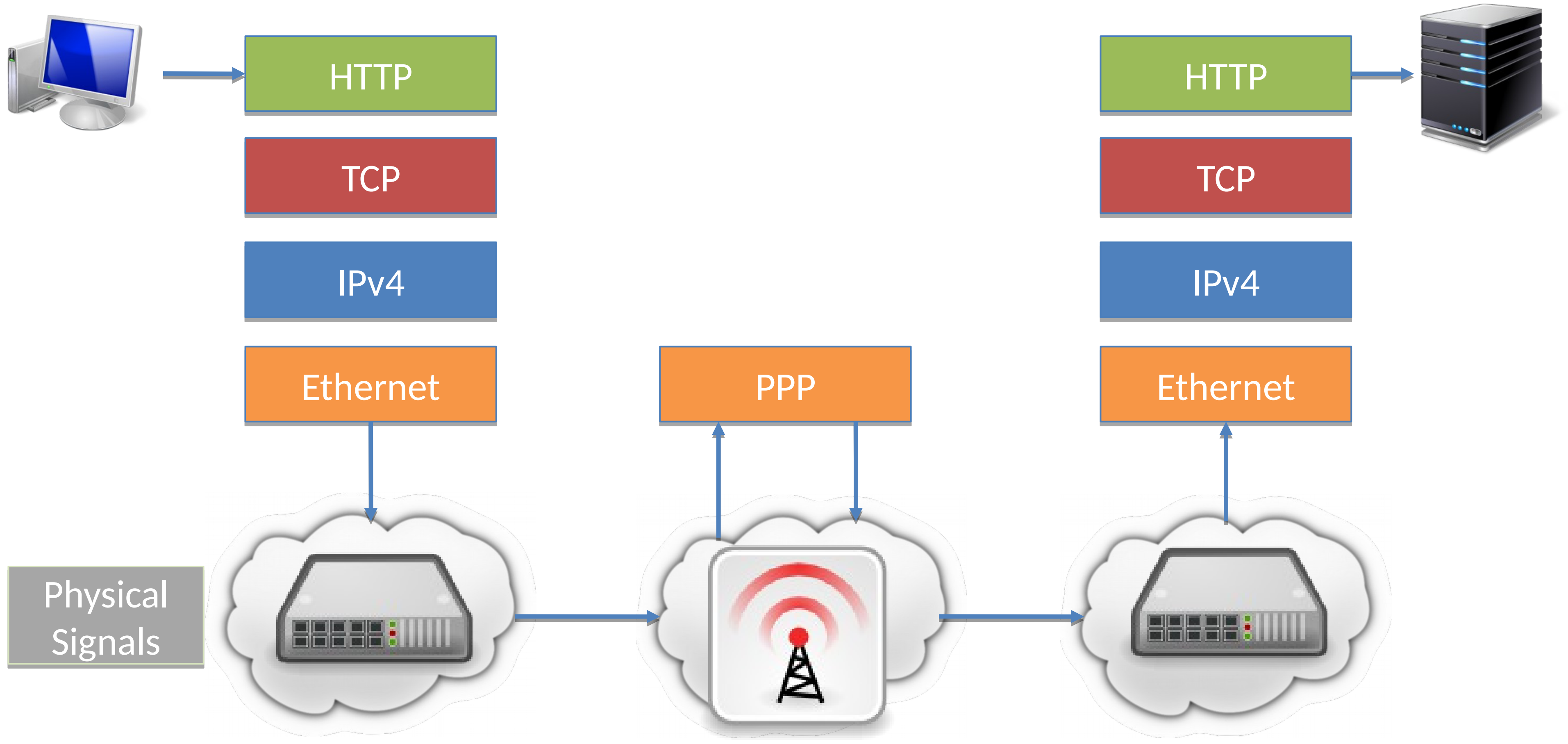
Network Security (2)

Computer Security 2 (GA02)

Emiliano De Cristofaro, Gianluca Stringhini

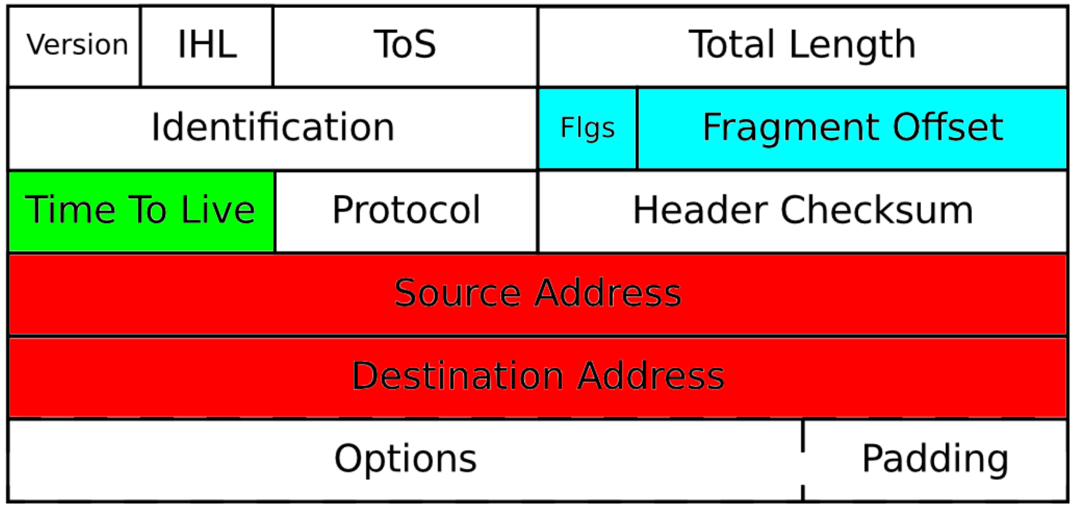
Thanks to Giovanni Vigna
for some of the material

Some Recap


HTTP
TCP
IPv4
Ethernet
PPP
HTTP
TCP
IPv4
Ethernet
Physical Signals

Some Recap

Used to encapsulate data from higher layers – best effort, no reliability guaranteed
A few fields in the IP Datagram can be misused or have security implications



- The Time to Live (TTL) Field can be used to mount traceroute attacks
- Other fields that are security related:
  - Flags + Fragment Offset – IP datagrams can be fragmented if the underlying link layer can handle only smaller packets
  - Source + Destination IP address – spoofing, hijacking

Ping of Death

Lower layer protocols can support packets smaller than the maximum IP datagram size (65,535 bytes). For this reason, datagrams can be split in multiple fragments

When a packet is fragmented:

- The datagram ID is copied in each fragment
- The “more fragments” flag is set except for the last fragment
- The “fragmentation offset” tells the position of the fragment in the sequence
- The “total length” field is the length of the fragment

Problem: the recipient operating system has to receive all fragments before it can reassemble them – the total datagram length is not known a priori

Ping of Death: total length can be higher than 65,535 bytes and can overflow a static buffer

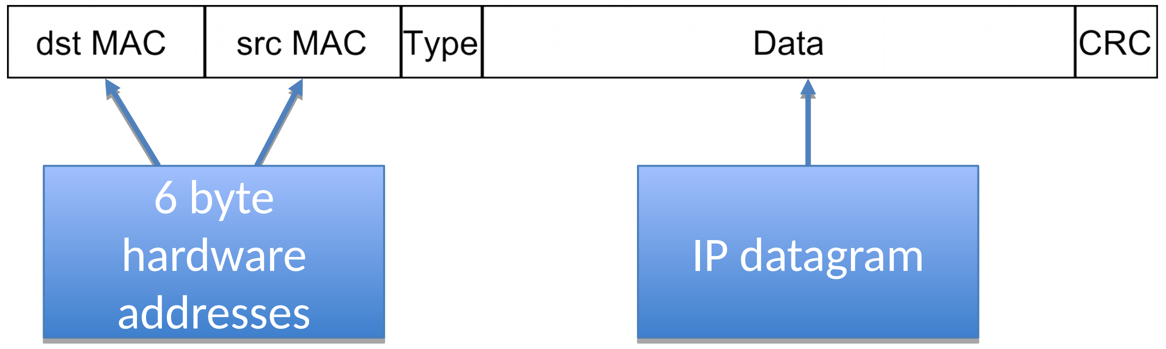
```
gianluca@console:> ping -l 65510 192.168.0.123
```

Now patched almost everywhere it used to affect most operating systems:
Solaris 2.4, MacOs 7, Windows 95, Linux 2.0

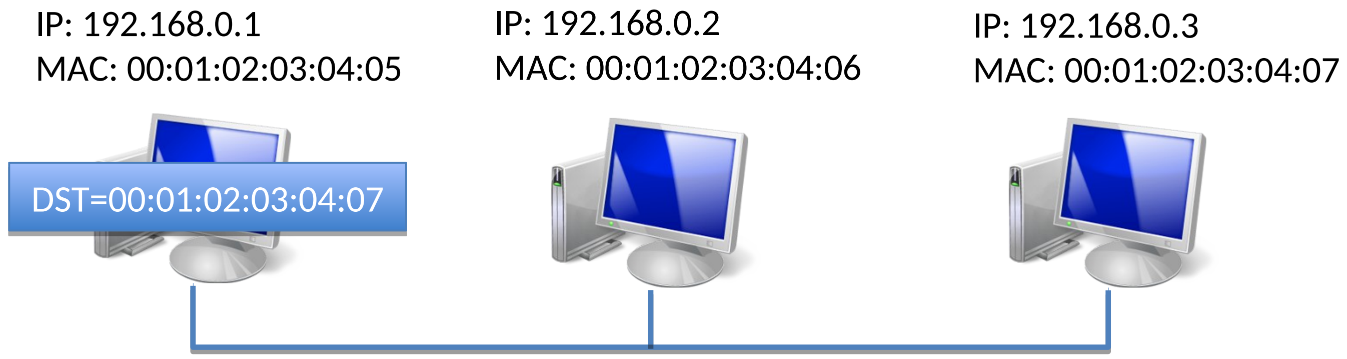
Good example of how standards ≠ implementation

IP: Direct (Local) Delivery

To reach computers in the same network (link), IP datagrams are encapsulated in Data-Link frames, the most common (among the wired ones) being **Ethernet**



Implements basic link functionalities such as collision avoidance and CRC

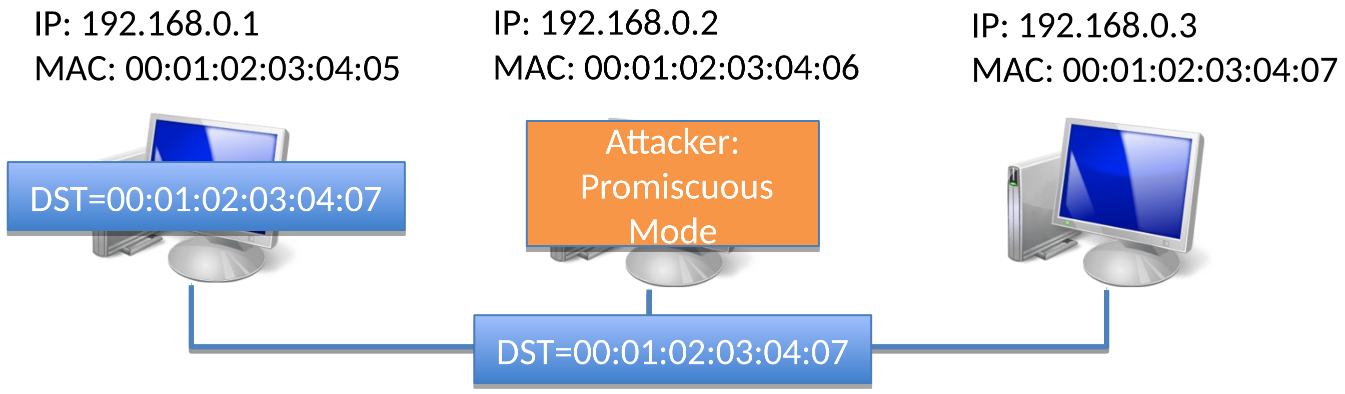


The network card passes to the operating system only those frames that are destined to that host (i.e., have its MAC address as destination)

Network Sniffing

The other hosts in the network "see" the traffic, but decide to discard it
Attackers can set their network card to *promiscuous* mode, and accept all traffic
→ The host becomes a **network sniffer**

Much of the traffic in the network is not encrypted: attackers can see it! (passwords, usernames, files)

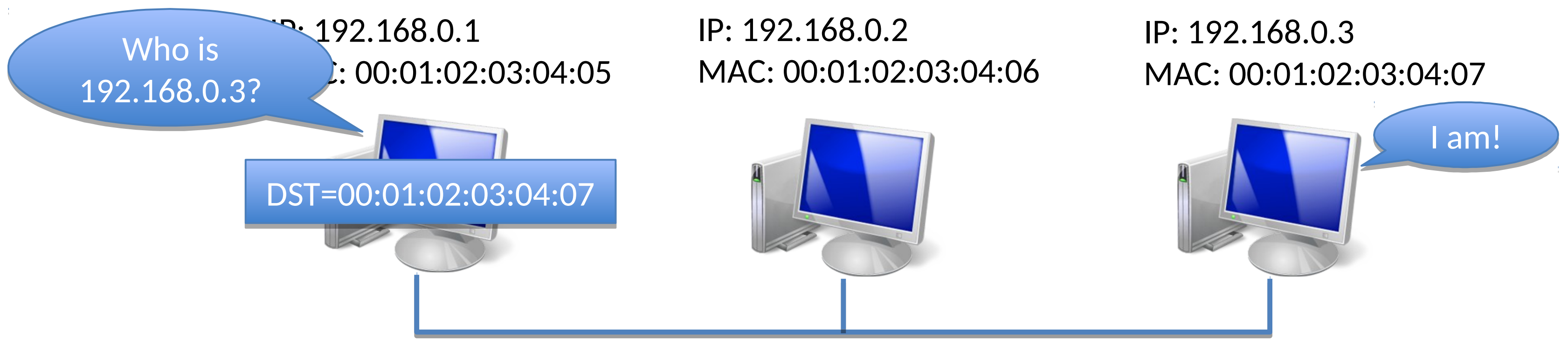


Sniffers on the network can be detected

- The operating system behaviour changes (some Linux kernels accept frames with the wrong MAC address but the right IP address)
- When in promiscuous mode, host latency increases (because it has to process more packets) – one can generate traffic directed to host 192.168.0.3 and observe changes in the network latency of 192.168.0.2 (the attacker)

The ARP Protocol

The Address Resolution Protocol (ARP) is used by hosts to know which MAC address is associated to which IP address on the local link



ARP Spoofing

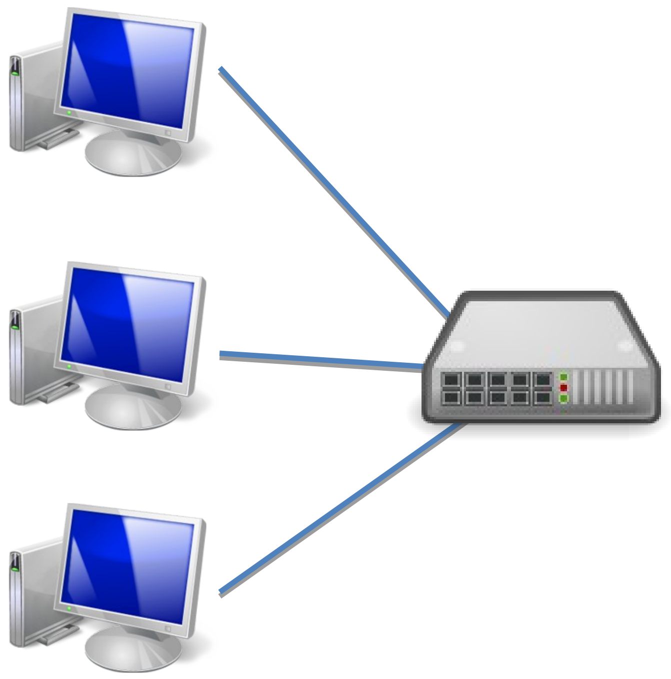
- ARP does not provide any authentication!!
- Attackers can “race” against the legitimate host and provide a fake IP / MAC mapping
- Traffic is redirected to the attacker

Issues with ARP Spoofing

- Race condition: legitimate ARP replies might restore the IP / MAC mapping
- Attackers continuously send spoofed ARP replies to keep the fake mapping

Switched Ethernet

There is no single link, a network switch has a point-to-point link to each host



- The switch keeps a mapping of which host (MAC address) is connected to which port
- The switch only forwards packets to the port to which the host is connected

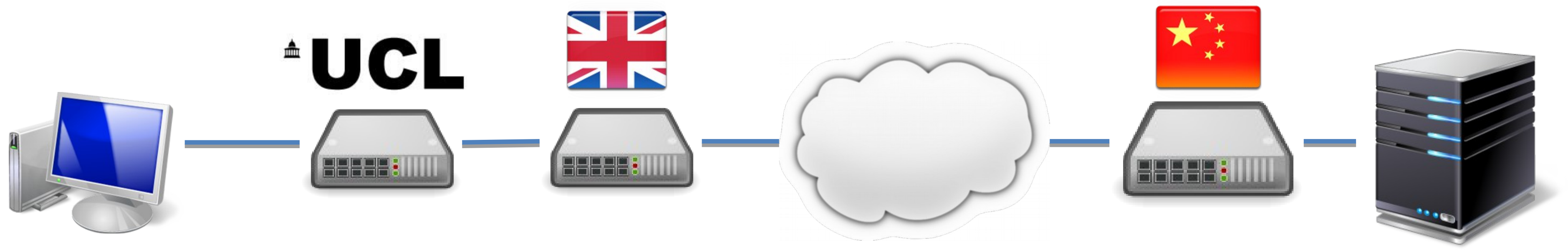
This mitigates some of the previous problems

- Sniffing is not possible (in principle)
- ARP spoofing is still a problem: attackers can make the switch believe that a host is connected to a different port
- **MAC flooding**: MAC / port mappings are stored in a table, if the table is filled some switches will start forwarding all traffic to all ports

The 802.1X standard provides some authentication: hosts need to authenticate to an authentication server before traffic is forwarded to them – one port, one MAC address

IP: Indirect (Remote) Delivery

Network routers know where to send datagrams based on their destination IP address



- Hosts send datagrams to their network gateway, which decides the next hop
- The gateway checks for a suitable router in a routing table and forwards the datagram to it
- If no suitable router is found, a “Host Unreachable” message is sent back
- Once the datagram reaches the destination network’s gateway, local delivery (e.g., Ethernet) is used

Routing tables can be set statically

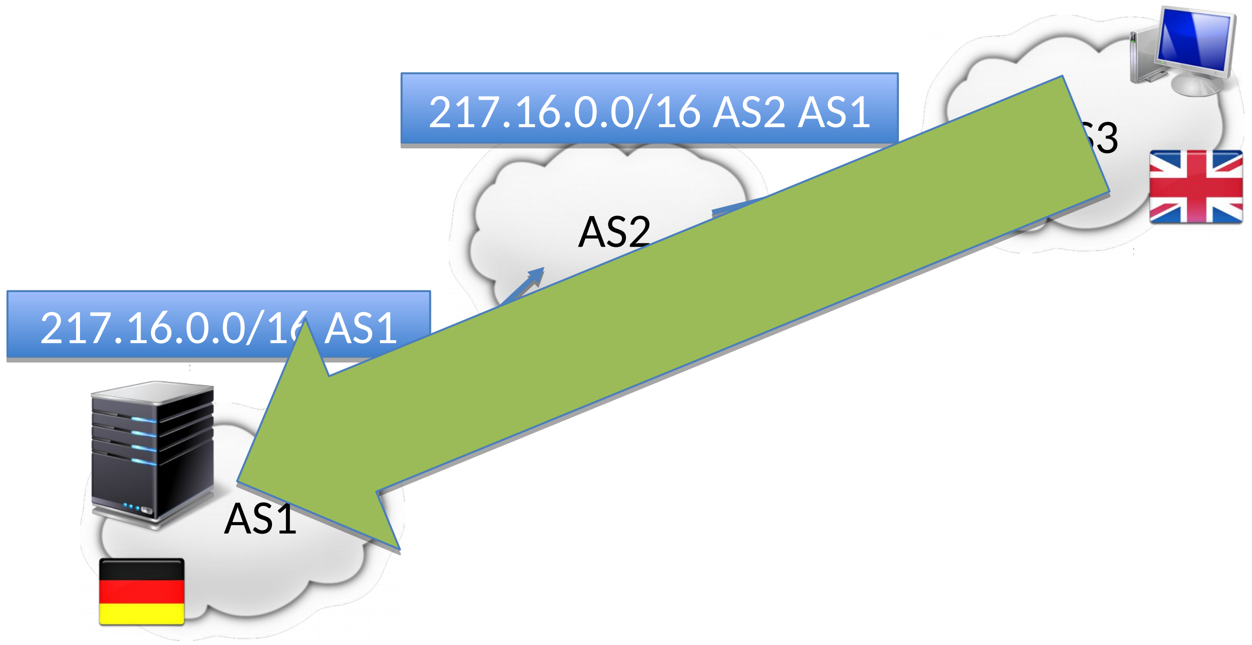
```
$  route -n
Kernel IP routing table
Destination Gateway      Genmask       Flags Metric Ref Use Iface
192.168.1.0 0.0.0.0      255.255.255.0 U     0      0   0   eth0
0.0.0.0     192.168.1.10 0.0.0.0       UG    0      0   0   eth0
```

Routing Algorithms

Allow routers to automatically negotiate routes **RIP, OSPF, IGRP**

The **Border Gateway Protocol** (BGP) is the protocol used by Internet Service Providers to tell each other which IP addresses they own → **BGP makes the Internet work**

Different networks on the Internet are called Autonomous Systems (ASes). Border routers advertise routes for the ASes that they control



BGP Hijacking

BGP routes can only be addressed by border routers, but they are not authenticated
If two routes conflict, the one that covers the smaller IP space wins

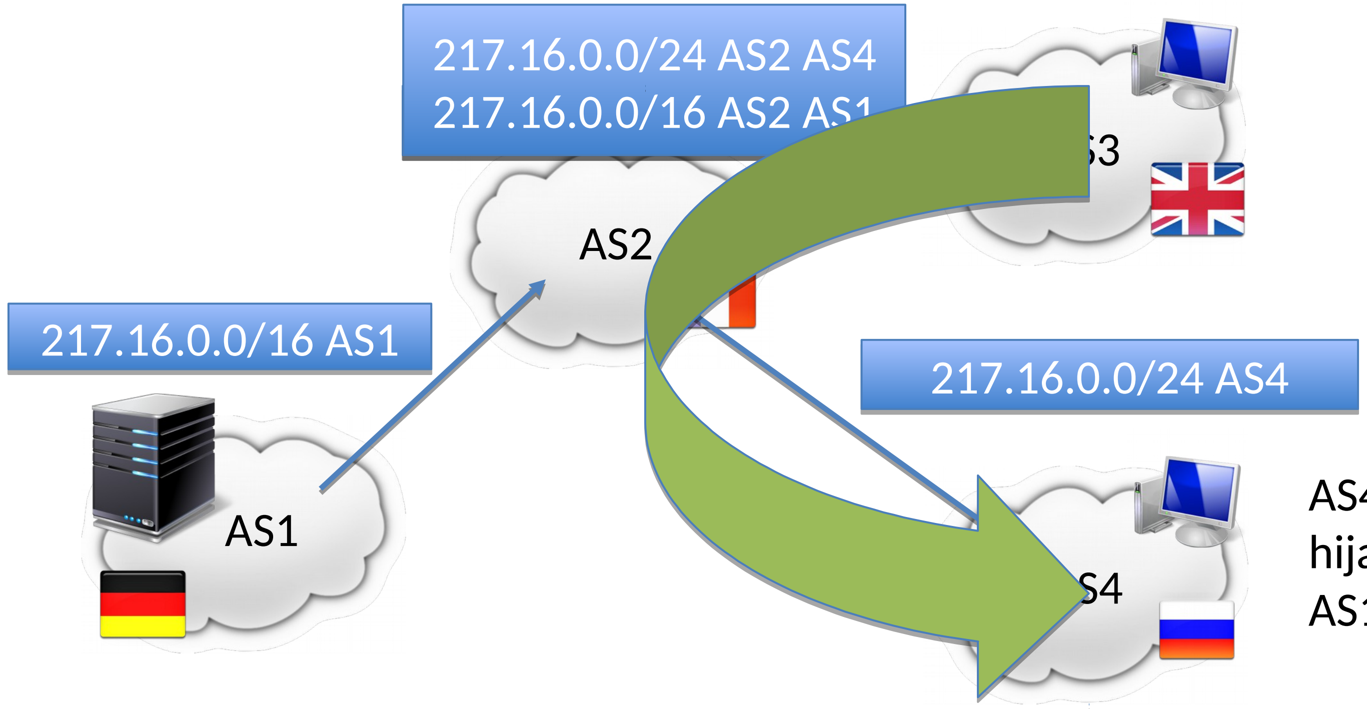
BGP routes can only be addressed by border routers, but they are not authenticated
If two routes conflict, the one that covers the smaller IP space wins

217.16.0.0/16 16 bits for network: 2^{16} hosts = 65,535 hosts
217.16.0.0/24 24 bits for network: 2^{8} hosts = 255 hosts

217.16.0.0/16 16 bits for network: hosts = 65,535 hosts
217.16.0.0/24 24 bits for network: hosts = 255 hosts

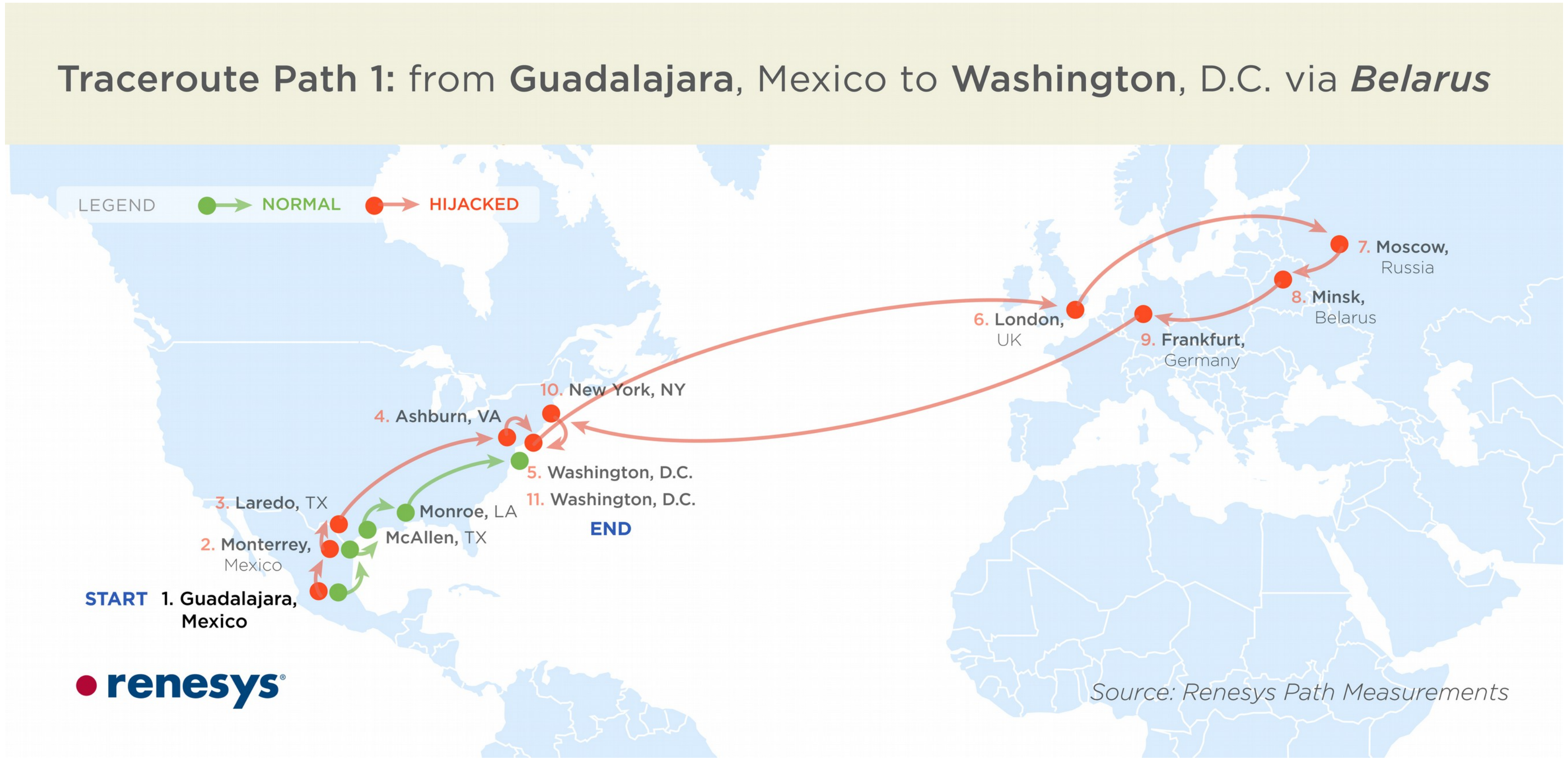
Rogue routers can use this to hijack BGP routes!

Rogue routers can use this to hijack BGP routes!



AS4 effectively hijacked part of AS1’s IP prefix

BGP hijacking: the problem is real



BGP relies on the trust between ISPs

Possible countermeasures to hijacking:

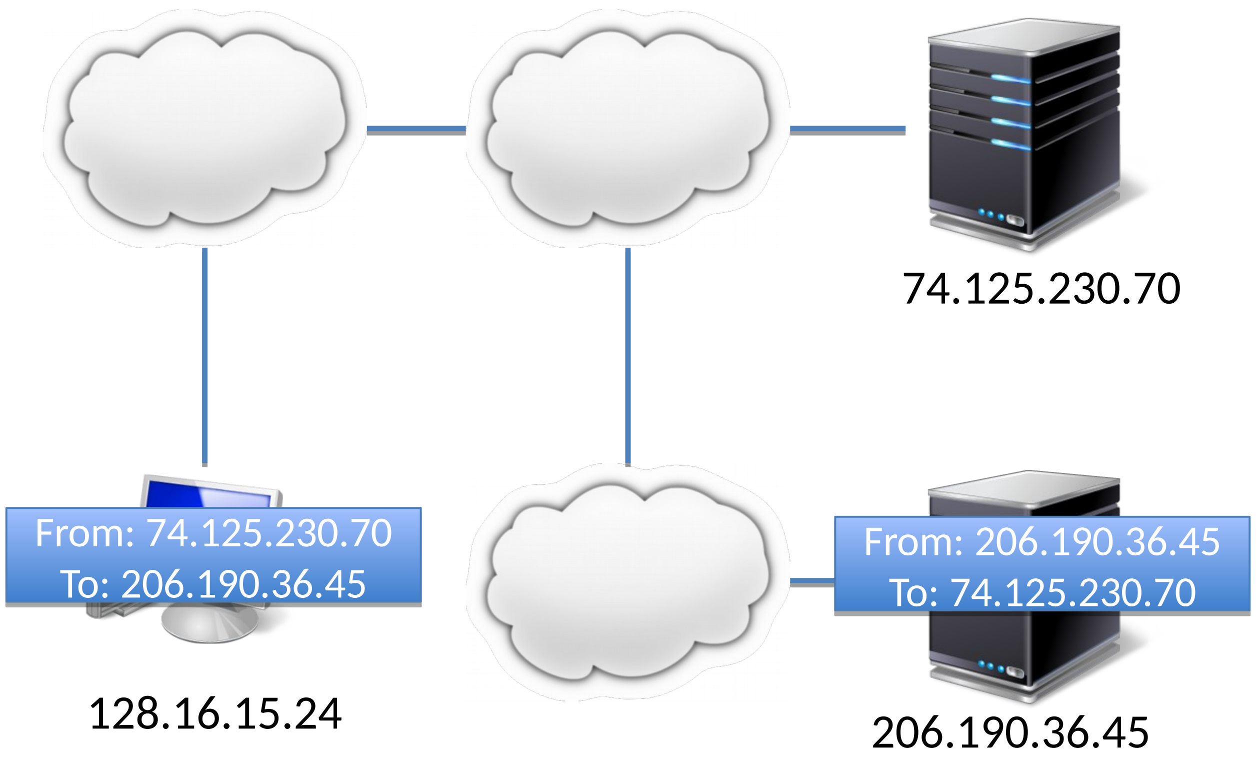
- Authenticating/signing BGP routes
- Having routers decide if an advertised route looks anomalous (arms race)

IP Spoofing

Attacker sends a message from a different IP address than the one he/she controls

IP spoofing in the local network → basically ARP spoofing

IP spoofing towards a remote network → **"blind" spoofing**



The recipient replies to the spoofed IP address

Technique commonly used in Denial of Service (DoS) attacks → more on this next week

Securing IP: IPSec

The IP protocol does not provide any authentication or encryption

IPSec is a suite of protocols that allows to **authenticate** and **encrypt** every IP datagram

Adoption is spotty because of legacy requirements, plus IPSec does not support Network Address Translation (NAT)

Key concepts:

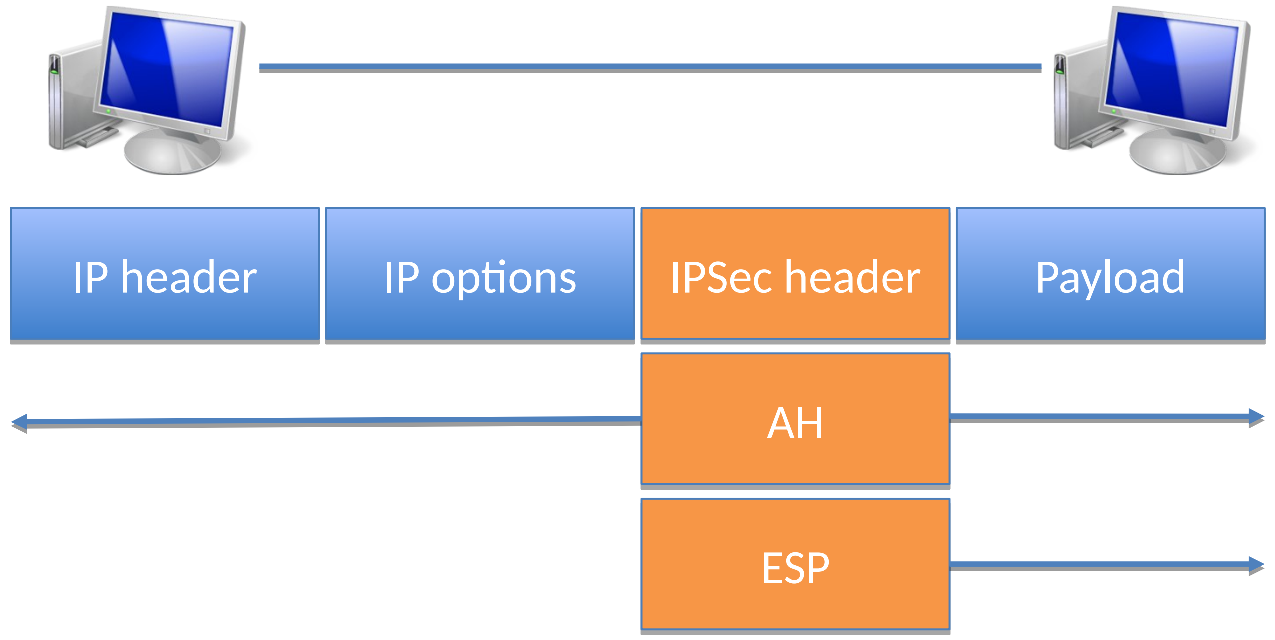
- **Security Association (SA)**: defined before communication is started (between endpoints) – specifies algorithms used for authentication and encryption
- **Authentication Header (AH)**: extension to IP that provides authentication and integrity of datagrams, a timestamp prevents **replay attacks**
- **Encapsulating Security Payload (ESP)**: extension to IP that provides authentication, integrity, and confidentiality (encryption)

Two operation modes:

- **Transport Mode:** host to host, IPSec is "added" to original datagram
- **Tunnel Mode:** gateway to host or gateway to gateway, original datagram is incapsulated in IPSec datagram → **allows Virtual Private Network (VPN) setups**

IPSec: Transport Mode

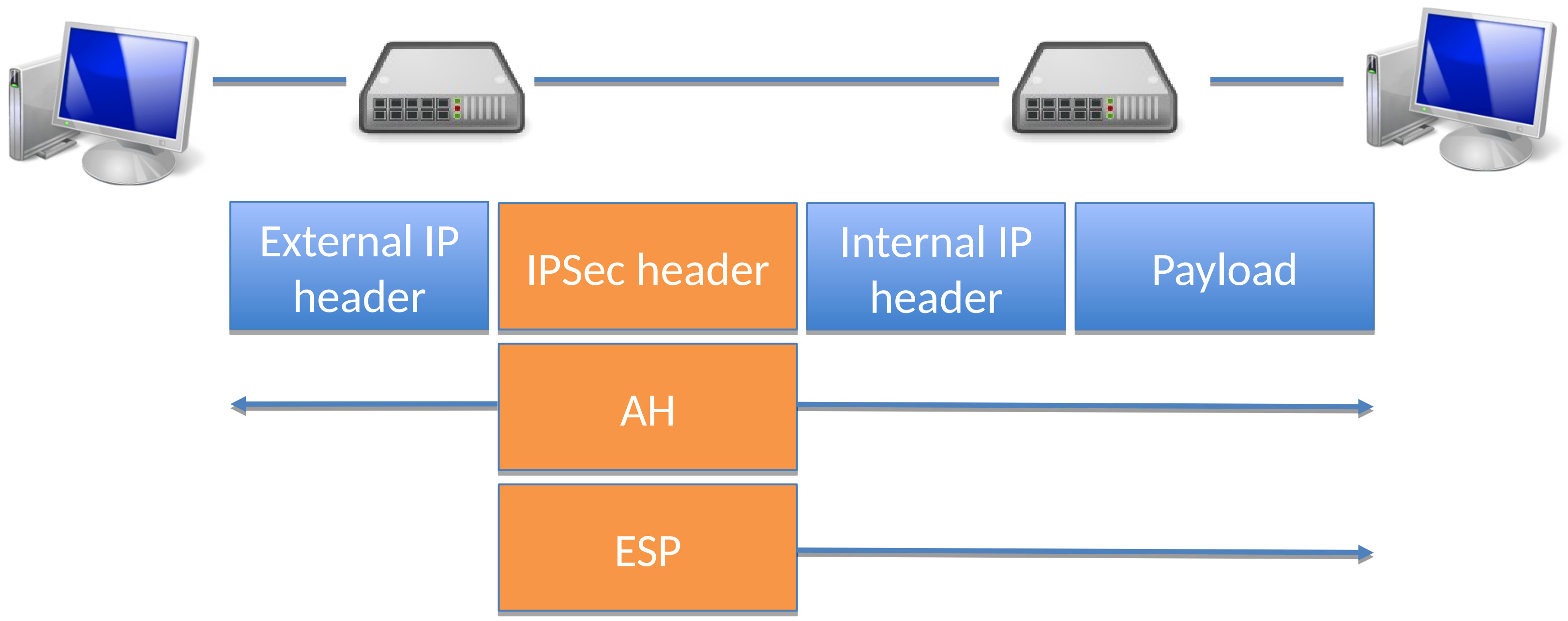
Connection is between two hosts



- AH can provide integrity (through hashing) and authentication (through a Message Authentication Code (MAC) protocol) to both IP header and payload – problem: IP options are variable and can change during transit – they are excluded
- ESP provides encryption (through symmetric algorithms), integrity, and authentication to the higher level payload only

IPSec: Tunnel Mode

Connection is usually between two servers, provides tunneling for Virtual Private Network (VPN) functionalities



- The two gateways incapsulate IP datagram into an additional IP header that is removed once the datagram reaches the other gateway
- The hosts in the network see each other as if they were in the same local network

The Next Generation: IPv6

- New standard created to overcome the IPv4 address shortage → IP addresses are not enough
- IPv6 uses 128 bit addresses
- Security is part of the standard (IPSec) and so is Quality of Service (QoS) and better routing
- Slow adoption always due to legacy requirements

New standard, created to overcome the IPv4 address shortage → 2^{32} IP addresses are not enough

IPv6 uses 128 bit addresses

Security is part of the standard (IPSec), and so is Quality of Service (QoS) and better routing

Slow adoption always due to legacy requirements

Transport Layer: UDP

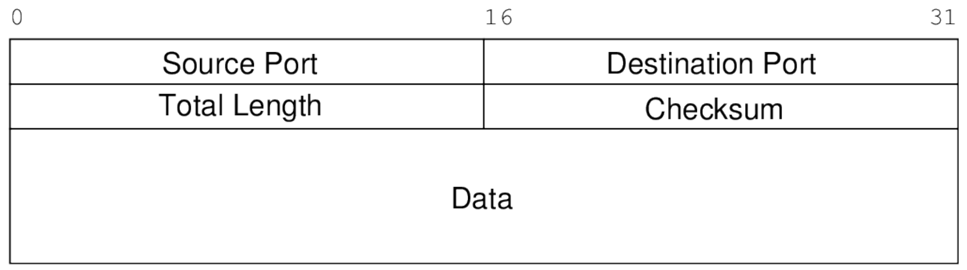
The IP protocol makes sure that traffic arrives at the destination host

Multiple programs (**services**) can be listening on a host, therefore the operating system needs to know to which program to forward traffic

Transport protocols → **each program is associated to a port (socket)**

User Datagram Protocol (UDP) is the simplest transport protocol

- Provides port abstraction to programs
- Messages are not acknowledged, reception is not guaranteed
- Ideal for applications in which losing some packets is not a big deal (video, audio)
- Services listen on a specific port, clients open a connection from a random port



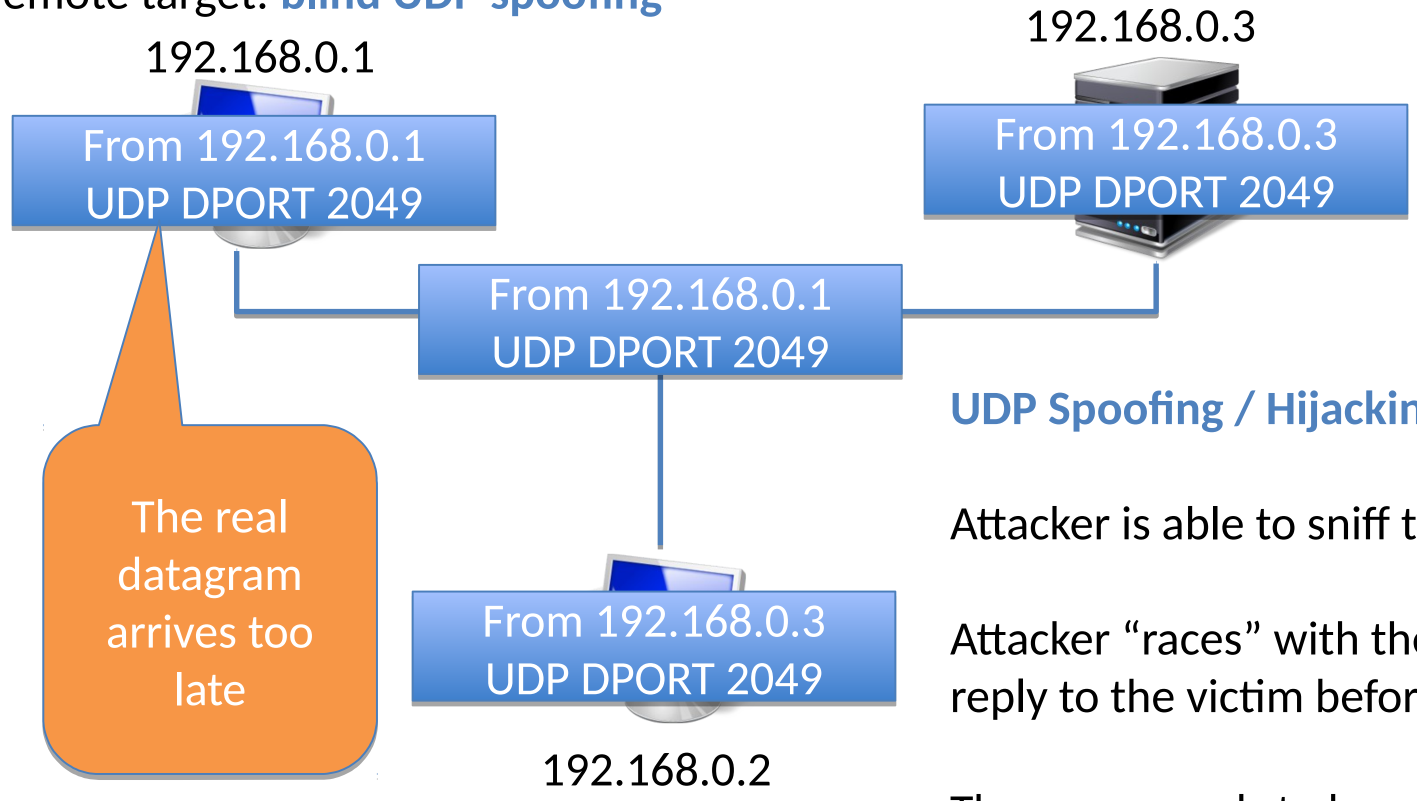
UDP ports: 65,535 possible
Popular protocols:

- Domain Name System (DNS) port 53
- Dynamic Host Configuration Protocol (DHCP) ports 67,68
- Network File System (NFS) port 2049

UDP Spoofing

Spoofing / hijacking a UDP stream is not any more difficult than doing it at the IP level
Two types of attacks:

- On the local network: **UDP spoofing / hijacking**
- On a remote target: **blind UDP spoofing**



UDP Spoofing / Hijacking

Attacker is able to sniff traffic on the network

Attacker "races" with the server to send a UDP reply to the victim before the server does

The same needs to happen for any subsequent datagram. Alternatively, the attacker can send the server offline (via a DoS attack)

UDP Portscan

To start an attack, miscreants need to know which services are listening on a host
The process of enumerating the listening services is called **portscan**

A UDP portscan is performed as follows:

- The person scanning sends an empty (no payload) UDP datagram to each possible port
- If no service is listening on the port, the server sends a **ICMP port unreachable** error message
- Some operating systems limit the rate of ICMP error message that can be sent, making these attacks slow (80 messages every 20 seconds $\simeq$ an hour to scan all ports)

Sweeping is the practice of scanning multiple hosts for a single port (for example to see which hosts on a network are running a webserver)

The legality of portscans is a sensitive matter

- In the UK, the Computer Misuse Act makes it illegal to even supply software that can be used to commit computer crimes!
- In any case, having written permission by any entity that you need to scan is important

Transport Layer: TCP

Uses the same port abstraction as UDP

Provides a connection-oriented, reliable stream delivery service, ensuring

- No loss of packets (called TCP segments)
- No duplication
- No transmission errors
- Correct ordering of segments

Connection endpoints establish a **virtual circuit** identified by source IP address, destination IP address, source port, destination port – source port is randomly picked by the client, destination port is the one the target service is listening on

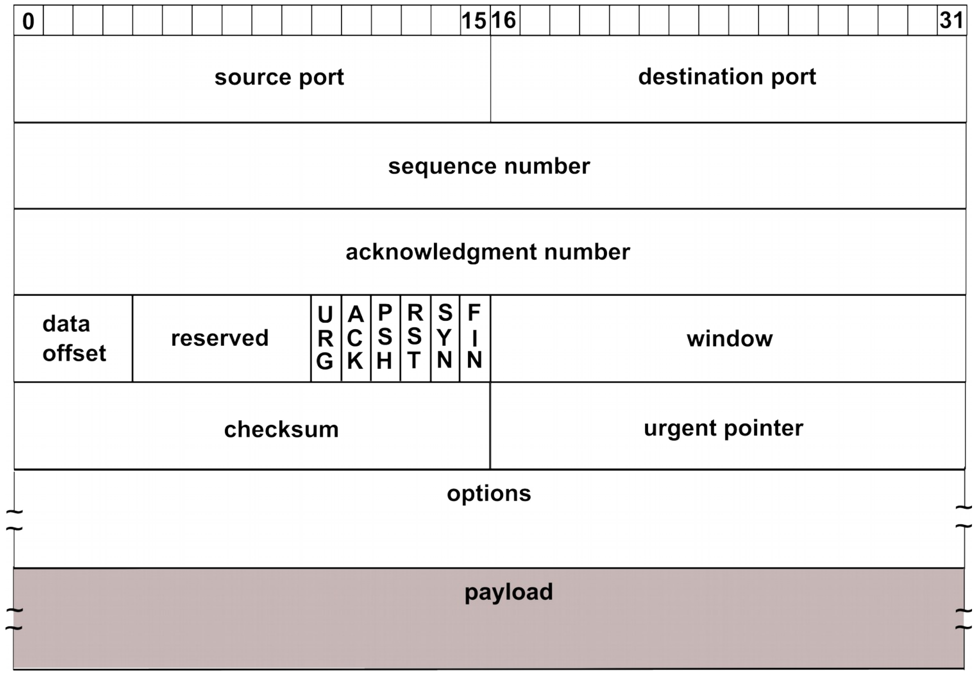
The TCP virtual circuit is composed of two streams (**full-duplex** connection)

Popular TCP services:

- HTTP port 80, HTTPS port 443
- SSH port 22
- SMTP port 25
- IMAP port 143

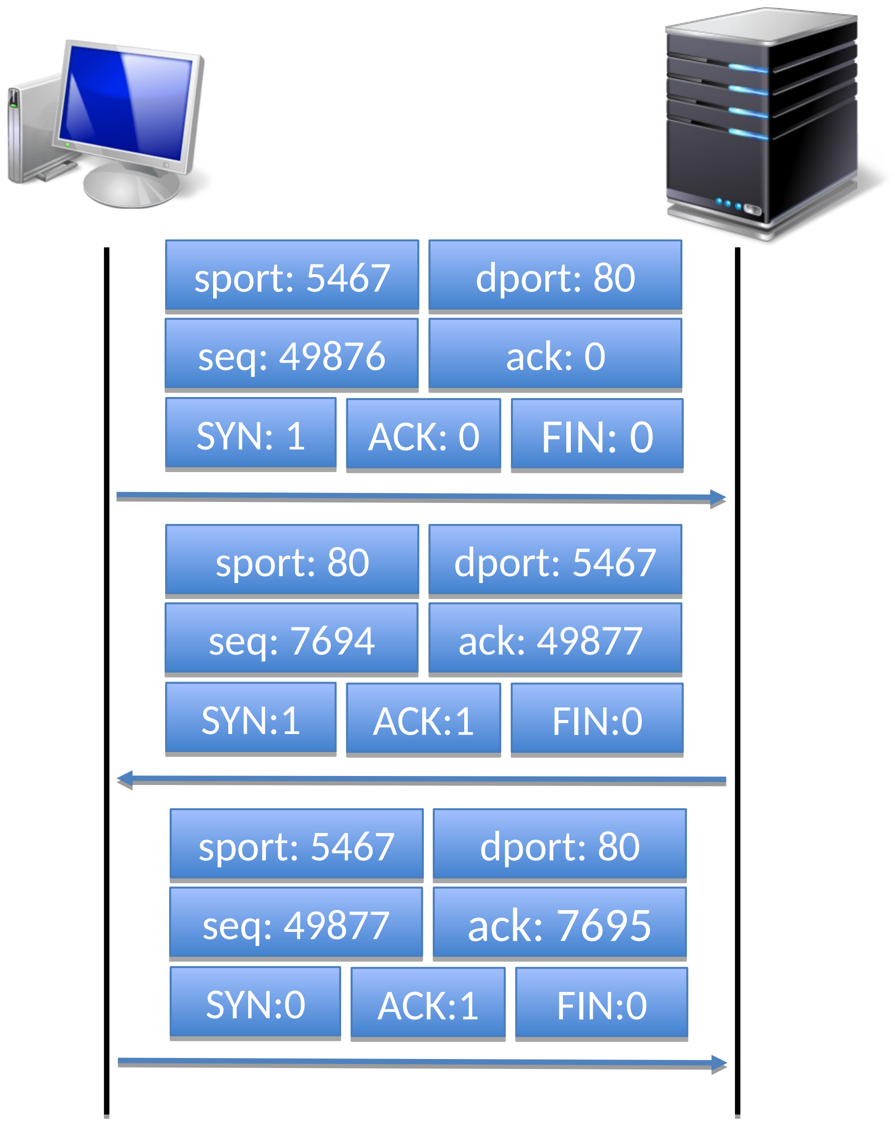
TCP Header

Sequence number: specifies the position of a segment in the communication stream
Acknowledgement number: specifies the position of the last segment received by the other host
Window: used to perform flow control – only segments within a certain window will be accepted by the receiving host
Checksum: checks the integrity of the segment



TCP Flags: are used to setup and shut down the virtual circuit
SYN: requests for synchronization of sequence and acknowledgment (ack) numbers
ACK: acknowledges the receipt of a segment
FIN: requests the shutdown of the circuit (gracious)
RST: requests to immediately shutdown the circuit (abrupt)
PSH: requests to pass the data stream to the user level (program) as soon as possible
URG: specifies that the data is urgent

TCP: Connection Setup



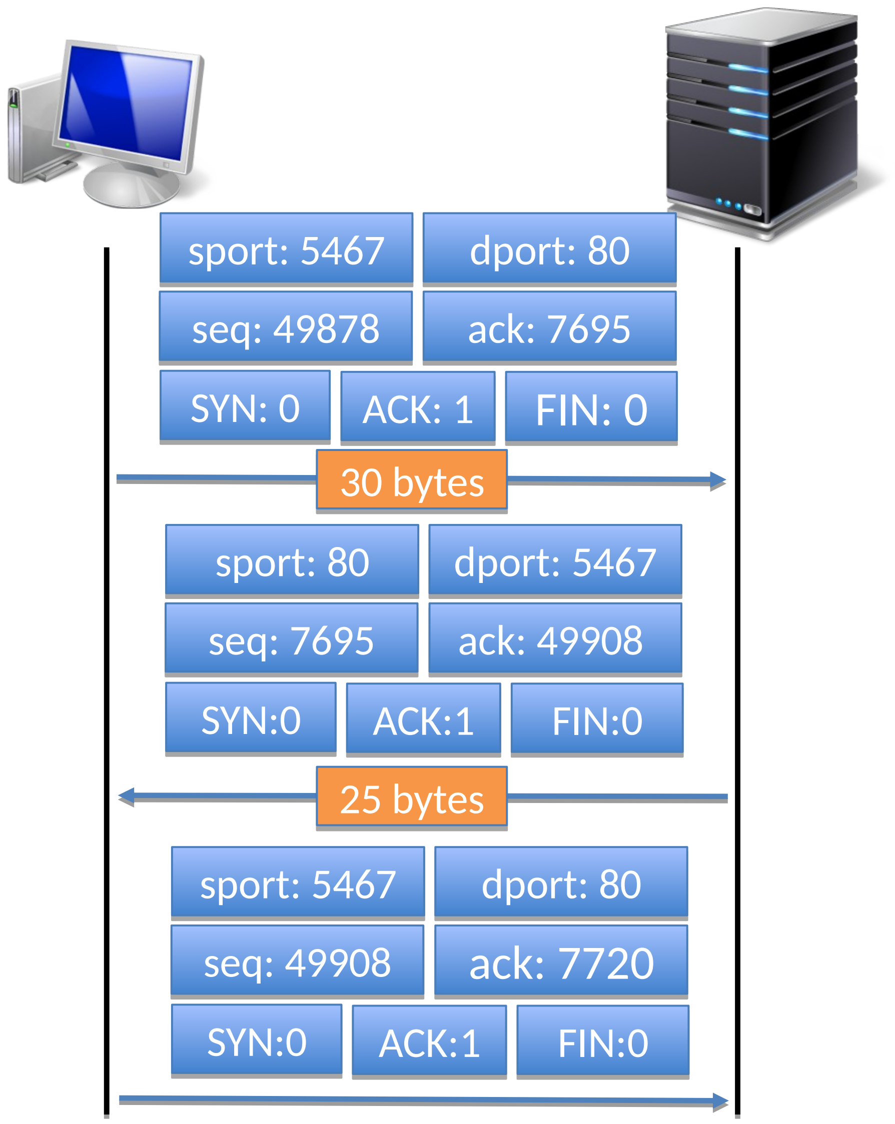
The client selects a **random sequence number**

The server replies with **a random seq number as well**

Three way handshake: three segments are exchanged before the connection is stablished

- **SYN**: the client communicates its sequence number to the server
- **SYN+ACK**: the server replies with its own sequence number, and sends back as ack number the number of the next byte in the sequence that it expects (client's sequence number increased by 1)
- **ACK**: the client sends the server's sequence number incremented by 1 as ack number

TCP: Data Transfer



Sequence and acknowledgement numbers are increased based on the number of bytes in the payload of the TCP segments received previously

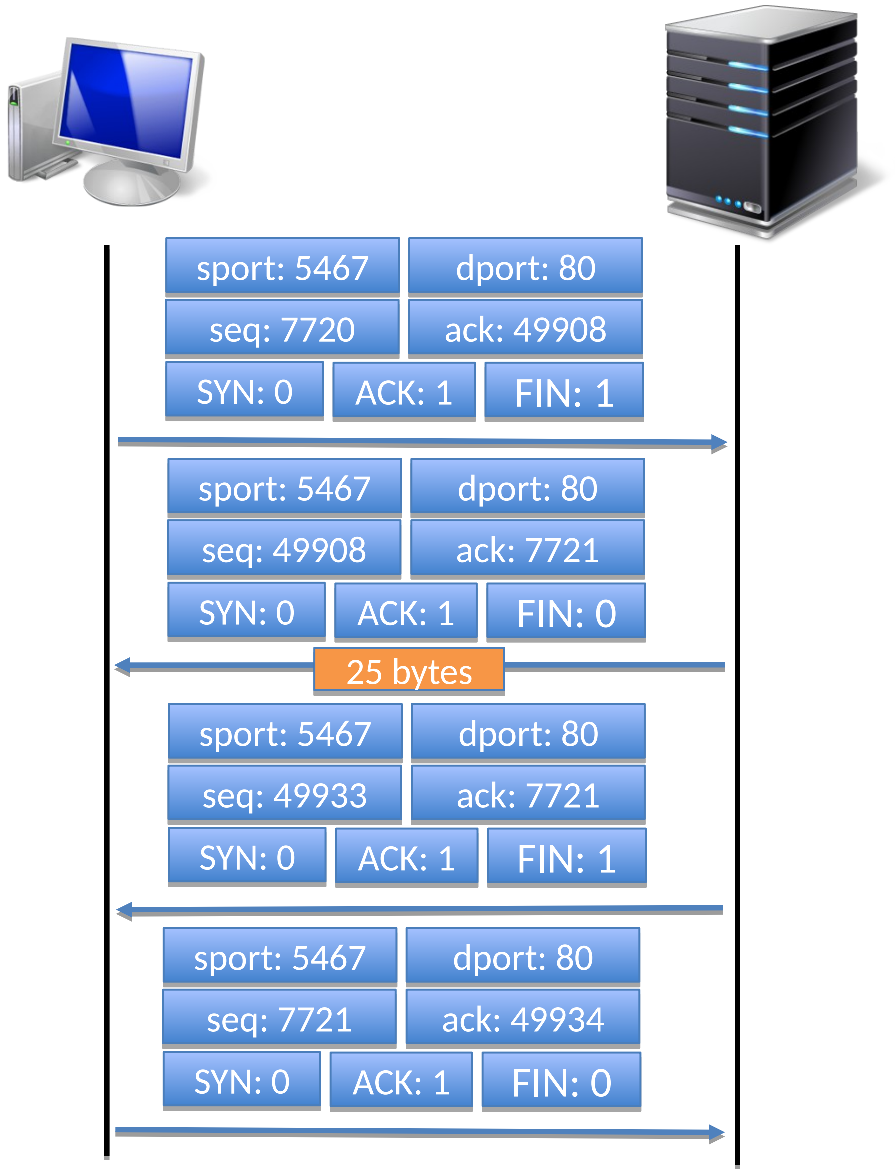
With each segment, hosts send back as ack number the next byte that they received, calculated as:

ack = seq + len(payload)

where seq is the sequence number of the other host received with the previous segment

In regular data transmission, **only the ACK flag is set to 1**

TCP: Connection Shutdown



Connections can be abruptly shut down by having one of the hosts send a segment with the **RST flag** set

However, the protocol offers a way to gracefully close the connection

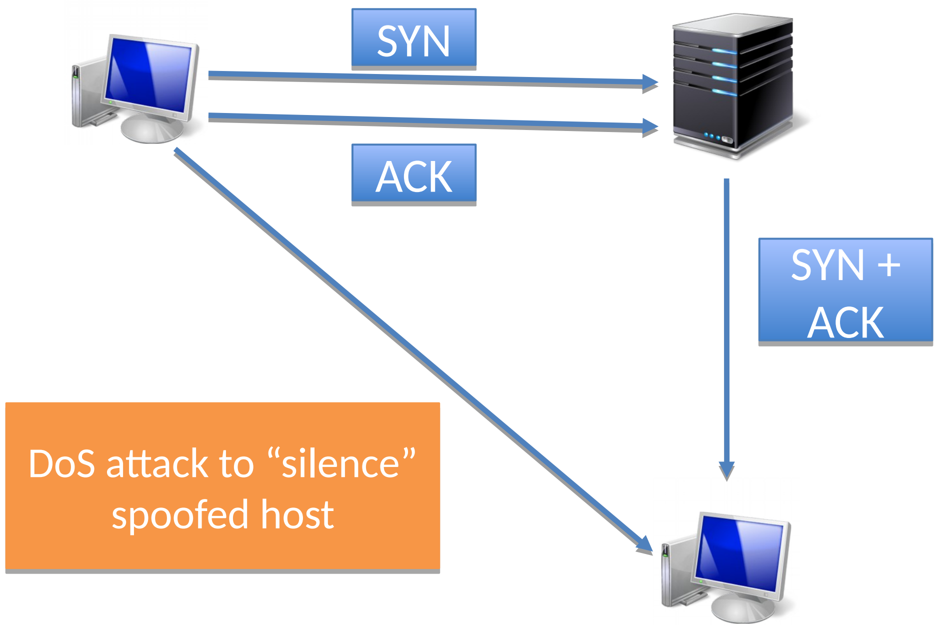
- One host sends a segment with the **FIN + ACK** flags set
- The other host keeps sending data until needed to finish the communication in progress, and finally replies with a segment with the **FIN + ACK** flags set
- When the other endpoint acknowledges the receipt of this segment with an **ACK**, the connection can be terminated

TCP Spoofing

Spoofing TCP is a lot more complicated than IP or UDP

- The spoofed host will start sending segments with the **RST** flag set when the victim host replies to it
- An attacker needs to have visibility of the other end's sequence number to successfully complete the 3-way handshake

Mitnick Attack used by Kevin Mitnick during his attack to the San Diego Supercomputer Center in 1994



Opening a connection from the spoofed computer allowed to login on the server with no password

The attacker overwhelmed the spoofed host with traffic so that it couldn't reply to the server's messages

Problem: The attacker needs to know the server's sequence number, two options

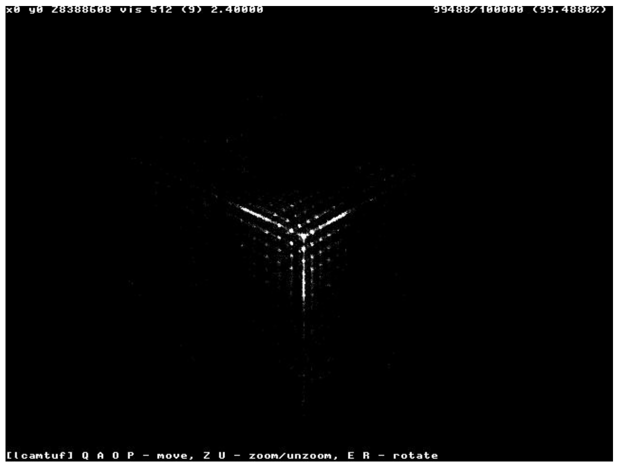
- Sniff it
- Guess it

TCP Sequence Number Problems

The security of the protocol relies on the fact that sequence numbers are as random as possible

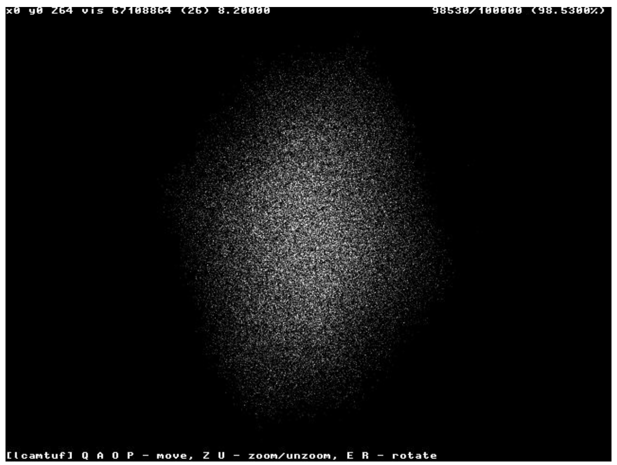
“Strange Attractors and TCP/IP Sequence Number Analysis” by Michal Zalewski shows that this is not always the case – most operating systems had flaws in the way they were calculating sequence numbers

Windows 95/98



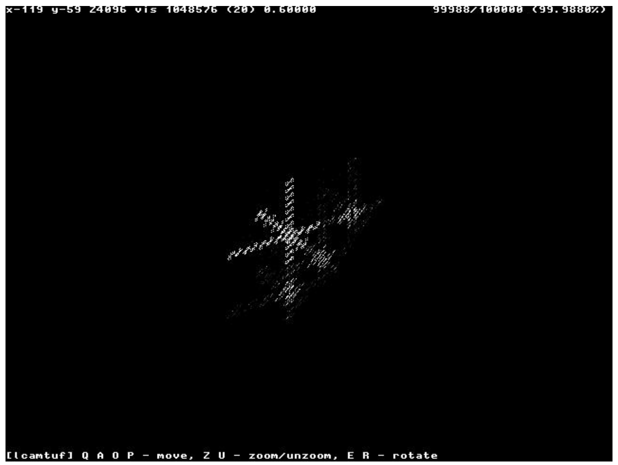
```
x0 y0 28388608 vis 512 (9) 2.40000                    99488/100000 (99.4880%)
[lcamtuf] Q A O P - move, Z U - zoom/unzoom, E R - rotate
```

Linux



```
x0 y0 264 vis 67108864 (26) 8.20000                   98530/100000 (98.5300%)
[lcamtuf] Q A O P - move, Z U - zoom/unzoom, E R - rotate
```

HP-UX

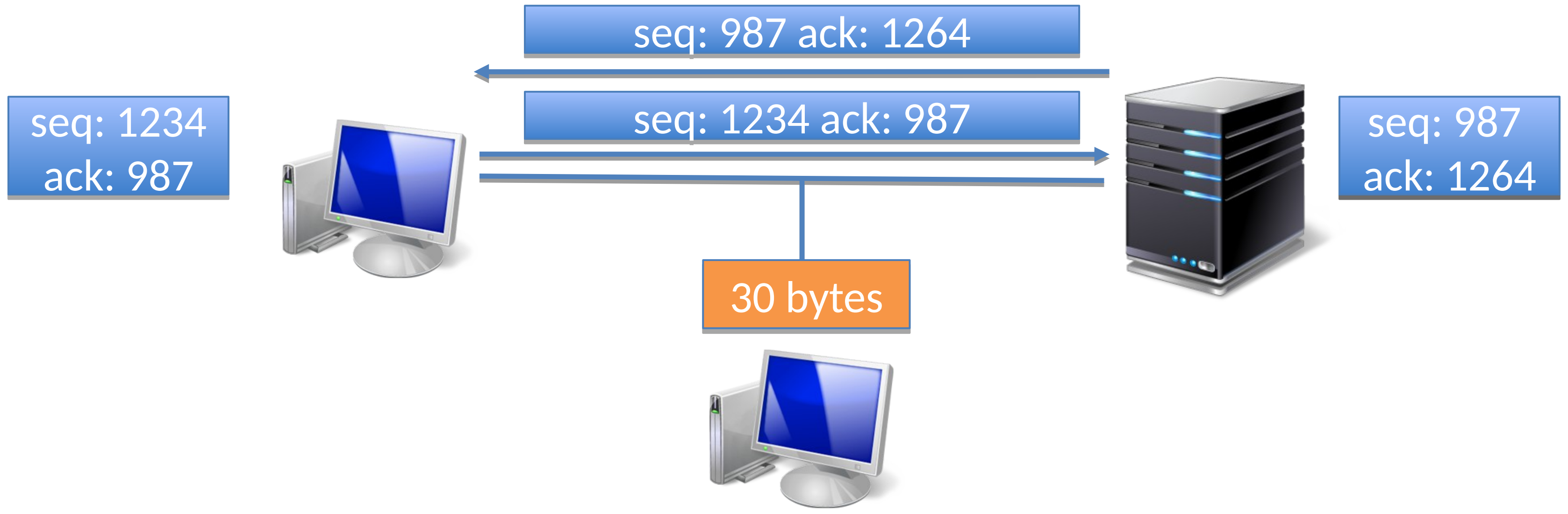


```
x-119 y-59 24096 vis 1048576 (20) 0.60000             99988/100000 (99.9880%)
[lcamtuf] Q A O P - move, Z U - zoom/unzoom, E R - rotate
```

Courtesy of Michal Zalewski

TCP Hijacking

Slightly different scenario: the attacker can see the communication between the two attacked hosts and wants to inject data inside an existing TCP circuit



- The attacker waits for the connection to be quiet – all segments have been acknowledged
- The attacker sends spoofed data of his choice to the server, desynchronizing the connection
- The server sends an acknowledgment to the other endpoint, but the sequence numbers don't match → an infinite loop of segments is generated – **ACK storm**
- The TCP standard says that segments with no data are not retransmitted if lost – at some point a segment will be lost and the connection is quiet again
- If the attacker injects more data, another ack storm will be generated

TCP Portscan

Same purpose of UDP portscan, but the higher complexity of TCP gives more options on how portscans are performed

Basic way of performing a portscan: **connect() scan** – the attacker completes the three-way handshake with the target host on the target port – if the handshake is successful the port is open

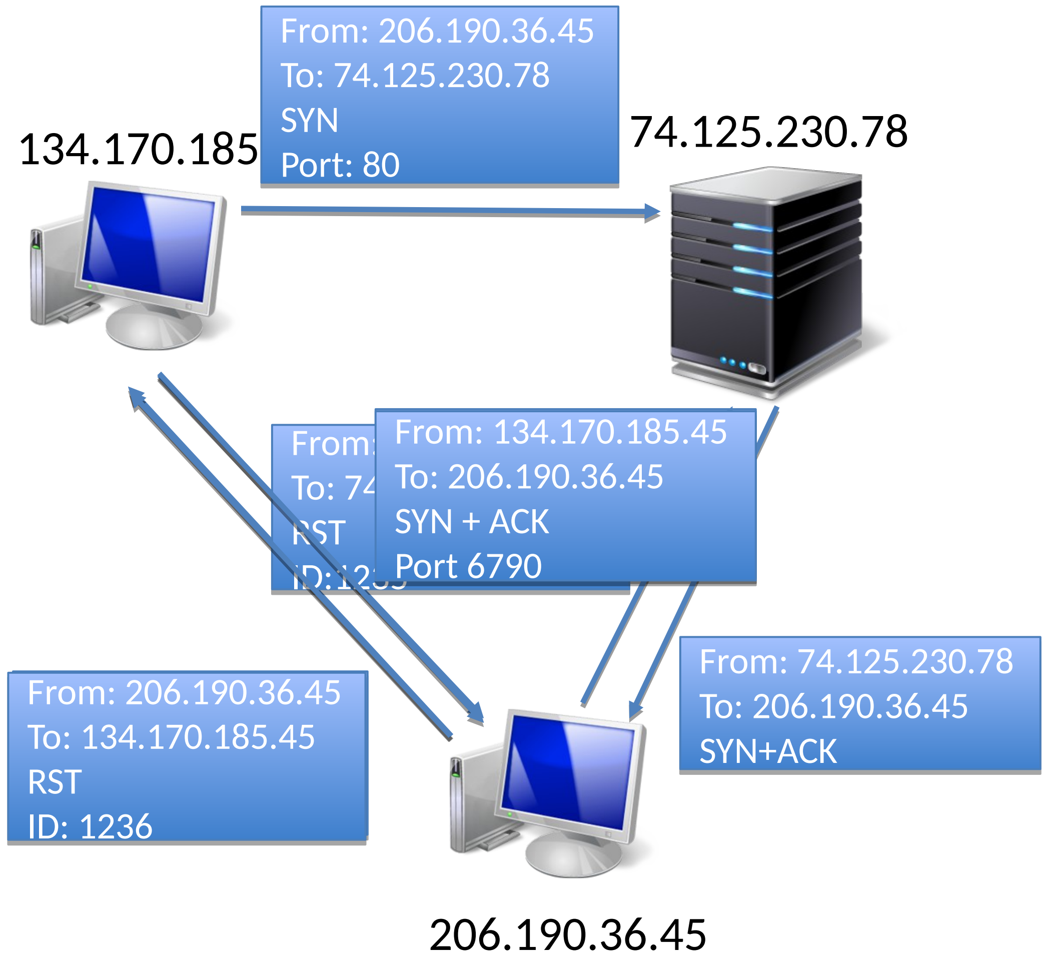
- It does not require root privileges on the host
- Noisy and easy to detect

Alternative: **SYN scan** (half-open scan) – the attacker sends a segment with the SYN flag set, if the port is open the host will reply with a SYN + ACK segment, otherwise with a RST

- Advantage: the connection is never open and it is not logged by the target operating system
- Detection relies on Intrusion Detection Systems – we will see them later in the class

Alternative types of scans:
FIN scan – a FIN segment is sent, if the port is open the segment is ignored, otherwise a RST is sent back – **unreliable**, if the segment is lost the port will appear as open
XMAS scan – segments with FIN, PSH, URG flags set
Null scan – segments with no flags are sent
Tool of choice for performing portscans: **nmap**

Idle Scanning



This type of scan is performed through a third-party **relayer**

The **IP ID field** of the IPv4 header is incremented by 1 for each datagram sent by the host

The attacker finds a host that is “idle” (no other traffic), and notes its current IP ID number

The attacker then sends a spoofed SYN segment to the scanned server, appearing as coming from the relayer

- If the port is open, the server replies with a SYN+ACK, the relayer send back a RST segment and its IP ID number is incremented
- If the port is closed, the server replies with a RST, which is ignored by the relayer

The attacker can test the outcome by checking the new IP ID number for the relayer

OS Fingerprinting

Different operating systems implement the TCP/IP stack in slightly different ways

By sending or observing **specifically crafted packets** it is possible to infer the operating system that is running on a host

Examples include:

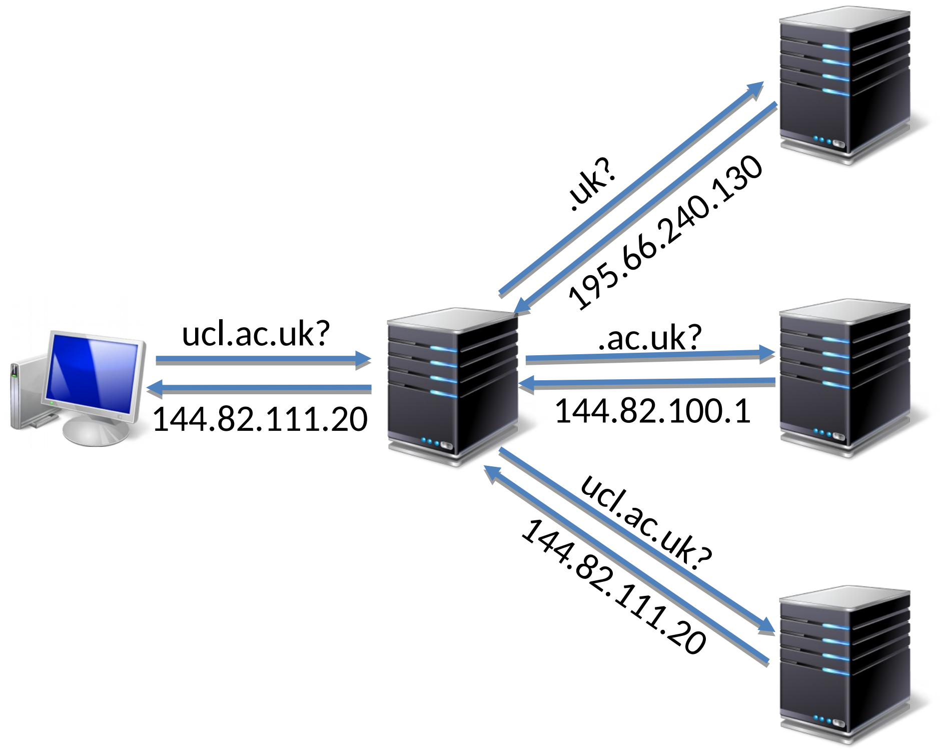
- Linux sets the initial IP Time to Live to 64, Windows to 128
- RFC793 states that hosts should not reply to a FIN packet sent to an open port, but some versions of Windows, Cisco ios, and HP/UX reply with a RST
- It is possible to find patterns in the TCP initial frequent number values of a host, and therefore infer its operating system
- Windows sets the IP ID field in host byte order instead of network byte order, therefore each new datagram increments the IP ID by 256
- Different operating systems set the Don’t Fragment bit in different cases, and this can be used for fingerprinting the operating system

Two ways of performing TCP OS Fingerprinting

- Active: by sending specific packets (tool of choice: **nmap**)
- Passive: by observing traffic and inferring the operating system (tool of choice: **p0f**)

The Domain Name System (DNS)

The Domain Name System (DNS) associates IP addresses to domain names that are easier to remember by humans



Two types of DNS servers:

- **Recursive servers** – clients make requests to them and they fetch the information
- **Authoritative servers** – know the IP – domain mappings

The DNS system is organized as a hierarchical tree: root servers → top level domain servers → second level domain servers etc.

Recursive servers will contact servers (starting from the highest server in the hierarchy) until they reach the requested domain

For performance reasons, DNS servers store records in a **cache** – most of the time they don't have to query the entire hierarchy

DNS: Security Issues

DNS is built on top of UDP – is affected by all the issues typical of UDP (e.g., spoofing)

If an attacker can spoof a DNS reply, he/she can redirect connections for other domains to his/her own IP address

To make spoofing more difficult, DNS requests include a **16-bit ID** that identifies them – clients and servers only accept replies containing an ID that they recently issued

DNS Cache Poisoning

Some DNS implementations will store in the cache any DNS record included in the responses they receive, without checking that such responses are relative to the queried domains

```
;; ANSWER SECTION:
ucl.ac.uk.      126     IN      A       144.82.111.20
google.com. 1234567     IN      A       192.168.0.3
```

Requests to Google are now hijacked!

To avoid cache poisoning, DNS clients must make sure that DNS responses are relative to the domain that they queried

DNS: The Kaminsky Attack

Goal: poisoning the cache of a recursive DNS server

Attacker needs to correctly guess the 16-bit DNS ID of a request and its UDP source port to succeed

- The attacker sends to the victim's recursive DNS server a number of DNS requests for domains he/she wants to hijack
- The recursive DNS server will issue DNS requests to the corresponding DNS authoritative servers – **attacker doesn't know source port and ID of these requests**
- The attacker sends a high number of spoofed DNS replies to the recursive server, hoping to guess the correct source port and DNS ID – if the attacker wins the race with the legitimate server, cache is poisoned

Problem: **there are many source port – ID combinations**

The attack leverages the birthday paradox: if a random function yields results in a range H with equal probability, after evaluating $1.25\sqrt{H}$ elements there is a 50% chance that two of the previously observed elements were the same – in practice the attacker has to perform less tries than expected

Example: In a room with 24 people there is a 50% probability that two people have the same birthday

In the DNS ID case: $1.25\sqrt{65,535} = 320$ tries on average

Solution

DNSSec – DNS Security Extensions servers sign records with their private keys, prevents spoofing and poisoning – requires a re-deployment of the entire DNS infrastructure, which is problematic

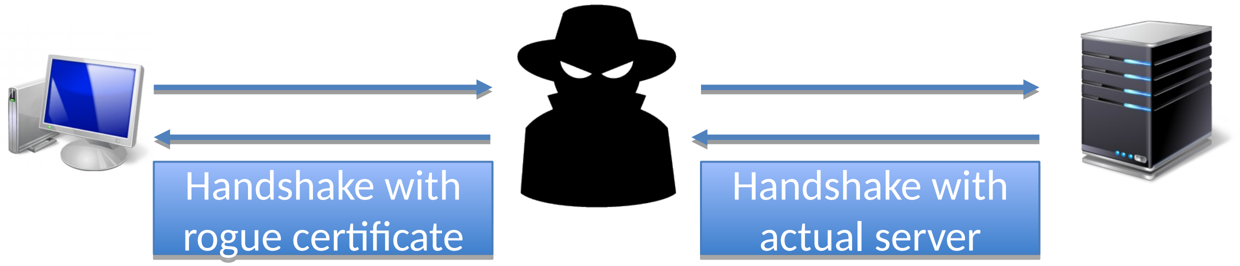
Secure Sockets Layer (SSL)

Secure Sockets Layer (SSL), together with Transport Layer Security (TLS) are the most common protocols to provide encrypted communications over the Internet. Commonly used on the web (**HTTPS**), in email exchanges (**IMAPS, POP3S**), and much more

In SSL, connection endpoints authenticate each other with public key cryptography (by using certificates) and then negotiate a shared key to encrypt their communication (with symmetric cryptography)

SSL prevents spoofing and hijacking, because attackers can't read the shared key, which can only be decrypted with the other endpoint's private key

However, SSL Man-in-the-Middle is possible


Handshake with rogue certificate
Handshake with actual server

Browsers automatically trusts all certificates that are signed by trusted certification authorities (CAs)
CA certificates are stored in the browser

- The attacker installs a rogue CA certificate in the browser (for example through malware)
- The attacker performs a man in the middle attack, and pretends to be the target server – the client sets up a secure channel with the attacker (believing it is talking to the target server)
- The attacker can decrypt the victim's traffic, read/modify it, and then relay it to the target server by opening a new SSL channel

File Transfer Protocol (FTP)

Simple protocol used to retrieve and upload files on the Internet

Peculiarities: FTP is an **out-of-band control** protocol – control traffic and data are exchanged over two different connections (using two different TCP ports!)

- Control stream: uses TCP port 21 on the server
- Data stream: two modes of operation
  - **Active mode**: the FTP client opens a random port and sends it to the server, which connects back from TCP port 20. Data is then transferred
  - **Passive mode**: the FTP server opens a random port and sends it back to the client, which connects to it. Data is then transferred

  Active mode has problems in the presence of NATs, passive mode has problems in the presence of Firewalls

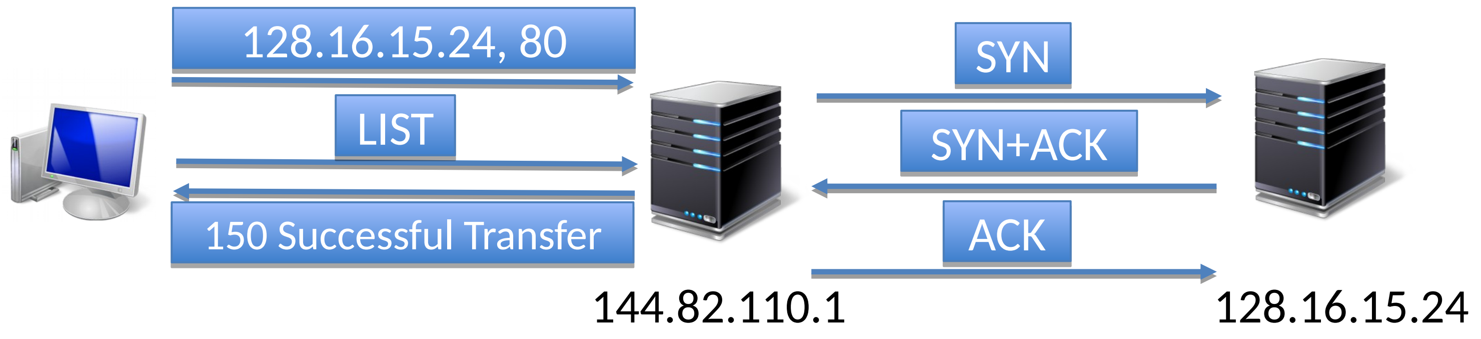
FTP is not encrypted, in particular usernames and passwords are sent in the clear
(FTPS introduces **encryption**, but this **creates problems with firewalls** that don't know the ports negotiated between client and server)

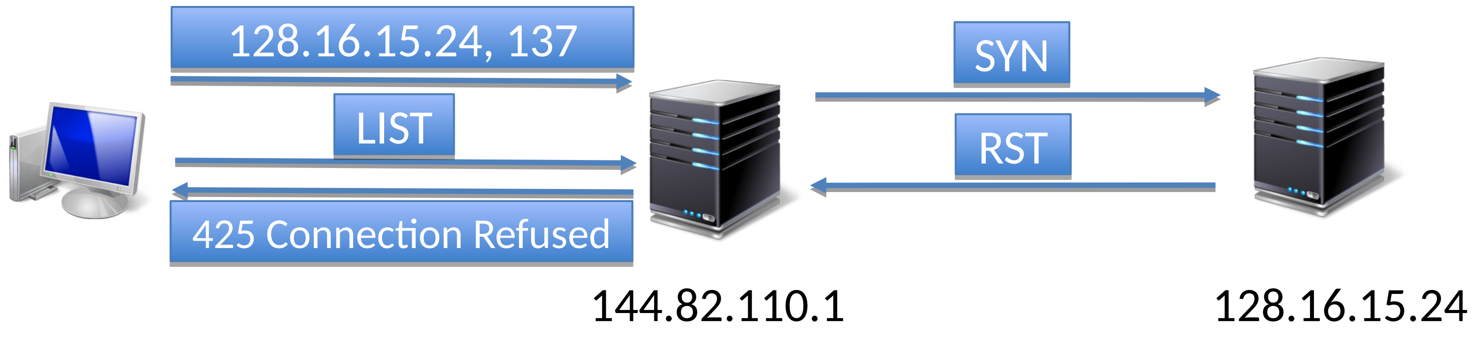
Vulnerable to sniffing, hijacking, spoofing

Anonymous FTP: used to access repositories
If misconfigured, attackers can read sensitive files (password files etc.)

FTP: Bounce Attack

In passive FTP, the client tells the server to which port it should connect to start the data transfer – **the port does not have to be on the same host that opened the connection!!**





FTP bounce attacks make it easier to evade restrictions on the source IP address:

- They can be used to scan hosts behind a firewall – more reliable than idle scan
- They can be used to send arbitrary data to a port

Telnet

Telnet is a service that provides remote command-line connection to hosts – developed in 1968

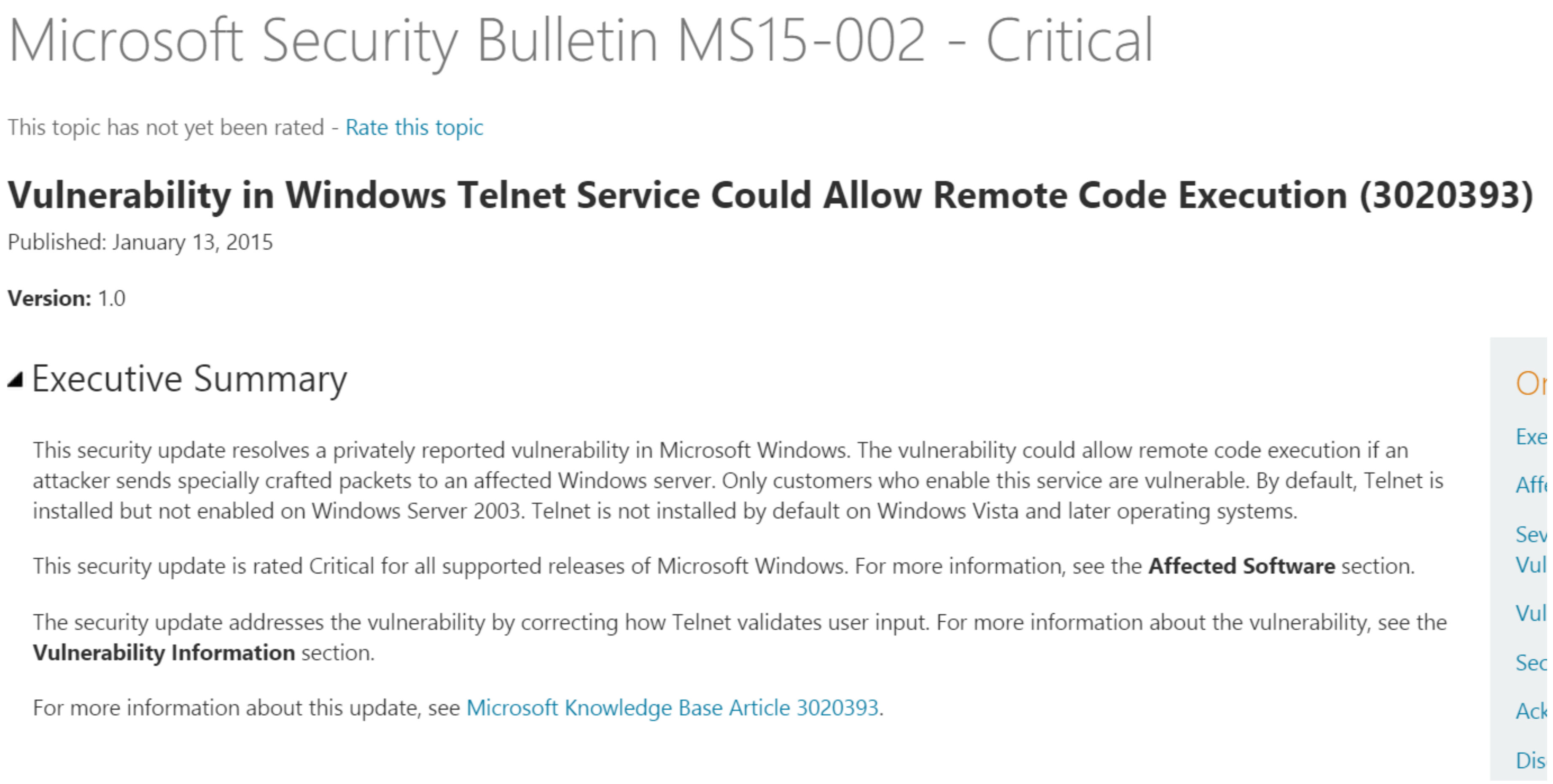
Multiple security problems

- **No encryption**, easy to eavesdrop
- **No host authentication**, easy to spoof

Telnet has nowadays been replaced by SSH, which is much more secure

From time to time, you still encounter network switches or appliances that still use telnet

Do You Think Telnet Is Outdated?



Microsoft Security Bulletin MS15-002 - Critical

This topic has not yet been rated - Rate this topic

Vulnerability in Windows Telnet Service Could Allow Remote Code Execution (3020393)

Published: January 13, 2015

Version: 1.0

◢ Executive Summary

This security update resolves a privately reported vulnerability in Microsoft Windows. The vulnerability could allow remote code execution if an attacker sends specially crafted packets to an affected Windows server. Only customers who enable this service are vulnerable. By default, Telnet is installed but not enabled on Windows Server 2003. Telnet is not installed by default on Windows Vista and later operating systems.

This security update is rated Critical for all supported releases of Microsoft Windows. For more information, see the **Affected Software** section.

The security update addresses the vulnerability by correcting how Telnet validates user input. For more information about the vulnerability, see the **Vulnerability Information** section.

For more information about this update, see Microsoft Knowledge Base Article 3020393.

Network File System (NFS)

Protocol that allows hosts to view, update, and upload files on a remote host

Main problem: **the only authentication used to be on an IP address basis** → open to spoofing

NFSv4 fixes this problem, introducing and mandating user authentication

Secure Shell (SSH)

Secure version of telnet

- Hosts are authenticated through public-key cryptography
- Communication is encrypted

SSH allows **tunneling**: other TCP services can be tunneled through SSH, providing them with the strong security properties offered by this service

Take Home Lessons

When thinking about the security of your network applications, **think about the security of the underlying protocols too**

Common problems:

- Lack of authentication
- Lack of encryption
- Poor implementations

"Fixing the Internet" is difficult because of **legacy requirements**

Some Reading

“Strange Attractors and TCP/IP Sequence Number Analysis” by M. Zalewski

“Simple Active Attack Against TCP” by L. Joncheray

“An Illustrated Guide to Kaminsky’s DNS Vulnerability”

(http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html)