Practical Network Defense

Notes of the lessons

Disclaimer:
These notes were produced for the Cybersecurity Faculty Practical Network Defense course taught by Professor Angelo Spognardi in Sapienza University.
The notes were made using the slides provided by the instructor and taking notations from the lectures.
These notes are not official, are not provided by the teacher, and may contain errors, grammatical or conceptual, being a personal revision of a student in the course.
These notes are not meant to be a summary of the course but an attempt has been made to aggregate all useful information (slides, explanations, etc...) related to the course.
Nevertheless, I do not ensure that they cover all course topics, partly due to the fact that they may change over time or because they were mistakenly not considered.

Author: Andrea Ciavotta
Sources*:* Lessons, Slides, Web

Summary

I. Recap on networks

- What is Internet and Internet Architecture

Internet is an interconnected network of networks.
This definition is a recursive definition. The idea is that there are several networks in companies, organizations, countries, ISPs, and we try to connect them together. To make it work, there is the need of hierarchy and hierarchical networks: we must be able to separate the duties.
The most clever intuition is hierarchy, in which we have to clearly define how to proper delegate duties; Internet works because there are many systems, devices, and networks working together in an organized manner so that there is a distributed workload.
We can distinguish several elements in Internet:
- the *Internet backbone* that is connecting ISP's backbones;
- the *ISP backbone* that is connecting organizations' backbones;
- the *organization backbone* connects LANs (local area networks);
- LAN connects end systems.

Backbone means the infrastructure composed by links that could be extremely far away and far extended; hundreds or thousands of kilometers in order to connect different devices.
The LAN connects end-systems. End-systems are devices that are able to generate and receive data packets; it is typically a device that is a host in a network.

It is not true that all the networks all connected to Internet; there could be also private networks. For example, sensitive organizations can have local networks that are private and are not connected to the Internet.
The communication on Internet is based on the Internet Standards.
A *standard* is an agreed set of rules and details that when followed can guarantee some properties and compatibility between different implementations of the same standard.
The Internet standards are tasks of different organizations like *IETF (Internet Engineering Task Force)* and related to the RFC Request for comments.
These organizations is composed by Internet experts that have the task to discuss standards, to spot weaknesses and problems of the proposals, to fix the problems and so on.

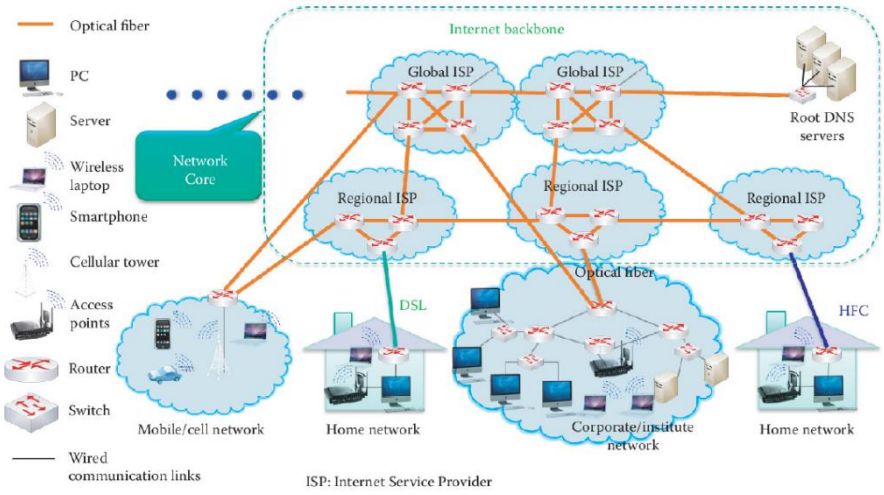


Figure 1: The Internet Architecture

The architecture of Internet is like the figure on the right.
There are several components that make the network to be alive.
There could be different physical implementations of LANs like 4G, 5G, Ethernet, Wi-Fi, DSL.
A LAN is made by devices that are considered to be in the same domain; they can reach each other without going out the network.
These are for example home networks, corporate network or mobile/cell network.
Those networks pay a subscription to an ISP (Internet Service Provider) that means to ask for receiving packets destinated to the network, to relay packets towards the destination of the packets, to receive requests for a offered service (if you are a server).
For example, a company that offers online services, may want to expose services online and requires some kind of services like a static IP address; instead an home user have different needs.
Internally, the ISP is connecting internal devices and this connection is called the ISP's backbone. If a user has to send a packet to a remote server, the ISP network has to move the packet up to a next hop and have to relay to other networks to forward the packet to the destination.
An ISP internal network usually is connected to others ISP's networks like Regional ISP or Global ISP.
The ISP has an internal network that is connected to other ISPs that could be regional/national ISP (Tier-2 ISP) or global ISP (Tier-1 ISP).
A global ISP is an entity can move data/packets without using and paying any subscription; there are 5 or 6 global ISP in the world.
The Internet Backbone connects tier-1 ISPs.
The IXPs (*Internet eXchange Points*) are access points, like big switches, where the backbones of tier-1 ISPs are interconnected.

The Internet backbone typically, and this means all the connections between regional and global ISPs, is wired-connected using the fiber optical connection (under sea, under the ground, in the buildings and so on).
Optical fiber is extremely reliable on the transmission of the signal, protected by electromagnetic fields and is very fast medium.
The Root DNS servers is an independent service, external to the ISP, that is made for giving service to the whole Internet and it is paid by ISPs.
In the Internet hierarchy we can distinguish between Network edge, Network core and Access networks.
In the network edge there are the hosts (servers, clients, P2P) and the applications are http, mail, Facebook, and so on.
The Network core is made by *edge routers* that connects an organization or ISP to the Internet, the interconnection of routers is done using optic-fiber.
The Access networks can be the connection of the edge points with the network core, using wired, or wireless communication links like Wi-Fi, 4G, 5G.
In Cisco naming we have the Access Layer, the Distribution Layer and the Core Layer.

The configuration of the Internet was reached thanks to the cold war.
This is because the Internet was supposed to be *extremely reliable*: if any node gets disconnected from the network (i.e. even if there is some failures in the network) then by using distributed algorithm, the Internet should be able to find a different path to reach the destination, so it is possible even in this case to connect every pair of hosts. The *mesh network* is the network where there are several different connections between two endpoints.

The routing mechanism can react to the dynamicity and changes of the network: routers can exchange information each other and they can update their knowledge of the network. In this way it is possible for the routers to work together to figure out the most efficient path for routing a packet from source to destination.
This kind of reactivity is possible thanks to the routing tables and the routing protocols.
There is a big difference between routing mechanisms and routing protocols, it is the same difference of drawing a map and reading a map.
The building and the update of the routing tables is the equivalent of drawing the map and it is done by the routing protocols: *RIP, OSPF, EGP* and so on.
Using the routing table is the equivalent of reading the map and there is one single rule that is the *Longest Prefix Matching*.
The calculation of the path for routing a packet is usually a local decision.

Routers don’t have a global knowledge of Internet; they only know the network how it was announced by their neighbors, so they have a partial knowledge of Internet. They only know some rules, and according to these rules, every router will decide to move the packet in different directions according to its rules.
When we talk about most efficient path, we are not talking always of the shortest path, but it depends on the network configuration: it is possible that a link has a huge cost respect to others, even if is faster, or more efficient.
Efficiency depends on the configuration decided by the network administrator.

- Protocol

A protocol is a set of specified rules that must be followed in order to make a service or something to work. It is made of:
- *set of procedure rules*: types and sequence of messages exchanged (format, syntax and semantics of packet), actions to take with respect to messages and events (request HTTP get with a given answer 200 OK for example).
- *message format*: format, size and coding of messages.
- *timing*: the time to wait between any event. For example, if an UDP packet is received, what is the time to wait before considering a packet lost.
In timing we have access to the medium specification, for example in Ethernet.
Also, in timing there is the flow control, that is typical in TCP protocol, like the sliding window that is possible to regulate according to how much is possible to send and receive packets, if it is possible to increase the traffic.

The *modularization* is used to split the functions that are expected to be done by a single protocol in several protocols. And it is possible to assume that there are different protocols that can perform the same task for example protocols TCP/UDP can do the same thing but in a different way. Also DNS can use UDP or TCP.
Every protocol usually works independently from the others: it does not deal with details of other protocols; it is supposed to work and that’s it.
Also, the layers can change without disturbing the other layers: you can, for example, use the same browser and changing the network from Ethernet to Wi-Fi without any problem.

The *packet switching* idea: in packet switching every packet is independent from the other; every packet can take a different direction even if it belongs to the same source and destination. They can adapt to the changes and the evolution of the network: packets of the same stream could take different directions each other.
The router can make a decision locally according to its rules, it does not know the future step of the packet that it forwards.
Therefore, routing is independent, and this idea is good for *best effort delivery* and

better for resource sharing. This makes possible also the network congestion and flow control.

- ## Ethernet (IEE 802.3) networks

The access layer is constituted by networks with endpoints of the same local management. Provides connectivity among stations on the same network.
Nodes in the same network can directly communicate among them: they can directly communicate each other without using any external device.
Inside the same LAN, the standard de-facto is the Ethernet family.
The used protocol is the Ethernet family; the structure of the packet used in Wi-Fi is the same of Ethernet.
The idea is that inside the same network every station can communicate directly with each other using a packet.
Each Ethernet packet is called ***frame*** and has a fixed format: what can change is the size of the *PDU* according of the type of technology.
Each host in an Ethernet network has a *Network Internet Card NIC* with a fixed address *Medium Access Control MAC*.
A **MAC Address**[1] is made by 48 bits or 6 bytes and ***uniquely identify*** hosts in the network: within one network we can have only one host with a given MAC and it is bound to the network interface.
If the host has *n* network interfaces, it is supposed to have *n* MAC addresses.
Each host only processes packets intended for it: network checks if the MAC address is the correct one, if it is not, it will discard the packet.
The correct visualization of this network is a unique shared cable that directly corrects every host in the same network: so, from this point of view, potentially any host in the network can receive a packet not intended for it.
This actually does not happen because using the switch there is a *segmentation of the network*, and the packet is not delivered to every host of the network, but only to the intended host using the destination number.

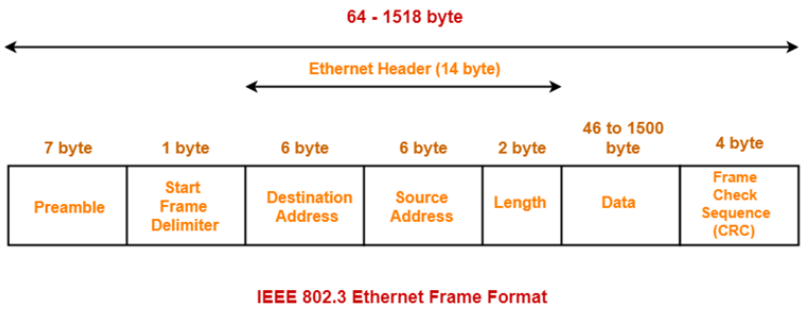


Figure 2: Ethernet Header

[1] *MAC Address*: https://en.wikipedia.org/wiki/MAC_address

The data, the real payload of the frame is the *Protocol Data Unit PDU* and it is variable from a minimum of 46 bytes to a maximum 1500 bytes.
The destination is the first group of 6 bytes and the source is the second group of 6 bytes. The preamble at the start of the header represents a really specific shape of the electromagnetic signal and it is used by the NIC to synchronize the clocks.

Repeating the idea, a local ethernet network LAN is a network where all the hosts are connected together with a "*shared transmission system*" based on Ethernet, and is it very similar as if they were connected to the same medium.
All the hosts in the same network can reach each other as if they were directly connected. It could be seen as:
- *logically*: if two devices are connected with the same single Ethernet cable;
- *physically*: many devices are connected with several Ethernet cables to a single device that typically is a *switch*, or *repeater* or *hub* or *bridge*.
The difference between bridge and switch is that bridge has one port for all the connected hosts and it forward packets to that port, it is then responsibility of the hosts to reject the packet if not directed to them.
Instead, the switch provides a segmentation of the network, and the packet is not delivered to every host of the network, but only to the intended host using the destination number and by detecting the single cable that is supposed to receive the packet.
In recent years, network hubs have slowly gone into obsolescence because they are considered less functional and secure than switches.

An Ethernet network constitutes a **broadcast domain**: when a packet is sent and it is supposed to reach all the hosts of the network, is it is said that the packet is sent in *broadcast*. All the hosts of the network are supposed to receive this packet.
The broadcast domain is the list of the hosts that are supposed to receive a packet that is sent in broadcast, and it corresponds to the list of all the hosts of the network.
For historical reasons also exist *collision domains*, that are not more used:
by using switches, we can avoid the collision because there is a dedicated cable only for every single connection in the network, so there is 1-to-1 connection without any risk of collision.

So, ideally, an ethernet network is a *single broadcast domain*, it means that every packet potentially could be received by all the hosts in the network: all the hosts received all the frames and only take the packets that belong to them.

But, in reality, the switches segment the network to limit the explosion of packets in the network: the packets are not really replicated everywhere but are only replicated in specific cables.
What happens is that the switch that receives the packet reads the MAC Address and replicates the message only in the specific port connected to the host.
Only the broadcast messages are replicated to all.

The *segmentation of the network* made by the switches is done in this way.
Every host is connected to the switch. So, every packet that a host generates is received by the switch; every packet has a source MAC Address and destination MAC Address.
In this way, the switch, learns and associate to that specific port with its source MAC Address. The switches remember the source MAC Addresses on the different ports. Then only replicate the frame on the segment where the destination MAC address is associated.
The information related to the association source MAC address and port number is stored in the *CAM Content Addressable Memory* table that is an extremely fast memory, for the switch. For the hosts MAC addresses are stored in the ARP table.

Why Internet is not a large Ethernet net?
Because actually it will be extremely inefficient for large networks, because Ethernet makes high use of broadcast packets.
Large networks must be split in order to reduce the broadcast domain.
There is the need a logical division of the networks: Ethernet is the Access Layer but there is the need of a Distribution layer.

The idea is that LAN are supposed to connect hosts locally and we need something that makes possible for a LAN to have access to the distribution layer (and so to other LANs) and distribute the packets.
There is the need of another protocol that is the *IP (Internet Protocol)* that is the protocol on which distribution layer is based.
The LANs are connected between each other using distribution layer, and the devices that makes this possible are the *routers* that use logical addressing.
Switches are only on the local networks; routers are the default gateways that give access to the Internet.
There’s an exception in which several routers are connected each other using a switch.

Difference between Ethernet address and IP

Ethernet has a physical address.
It is related to the NIC Network Interface Card, it is not possible (*not really, but usually it is not changed*) to change the MAC Address of the NIC of a device.
An Ethernet address is like the identity of a person, uniquely identify the person.
It remains always the same while moving on different networks.

IP is a logical address of the NIC Network Interface.
IP Address identifies the location and change by moving in another network; it changes according to where we are located.

An analogy to keep in mind the difference is this, imagine that you want to say something to somebody.
If both of you are in the same room (the same network) you can simply call his name (using the direct ethernet connection, so the MAC Address) and he will answer. If you are not in the same room, before sending the message, you have to know where the other guy is (IP Address).
The message has to leave the room through the door.
It is possible to know if an IP Address belongs to the same local network or to another network using the *subnet mask*.

- MAC Address

A media access control address (MAC address) is a unique identifier assigned to a network interface controller (NIC) for use as a network address in communications within a network segment.
This use is common in most IEEE 802 networking technologies, including Ethernet, Wi-Fi, and Bluetooth.
This 48-bit address space contains potentially 248 (over 281 trillion) possible MAC addresses.

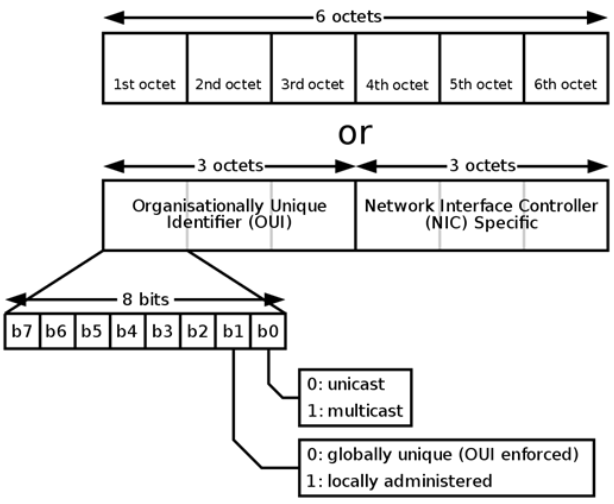


Figure 3: MAC Address Structure

The MAC Address is composed by 48 bit, divided into group of 6 bytes.
The first 3 bytes usually are the manufacturer of the NIC; so this makes possible to recognize the producer of the network card. The last 3 bytes are usually random.

- IP Address

Two versions of the Internet Protocol are in common use on the Internet today. The original version of the Internet Protocol that was first deployed in 1983 in the ARPANET, the predecessor of the Internet, is Internet Protocol version 4 (IPv4). The rapid exhaustion of IPv4 address space available for assignment bring to a redesign of the Internet Protocol which became eventually known as Internet Protocol Version 6 (IPv6) in 1995.

IPv4 defines IP Address with 32 bits organized in 4 octets/bytes (8 bits in each octet). IPv6 has 128 bits and uses hexadecimal digits.
For human readability, in IPv4 the bits in each octet are separated by dots.
Every IP Address has a certain amount of bits from the left that corresponds to the *network address* and the remaining correspond to define the *host* on the network.
All the hosts in the same network share the same network address, but with a different host part in the address.
The size of the network part is usually between 31 and 8, using the subnet mask it is possible to identify this division.

There are different types of IP Addresses:
- *Unicast* (one to one): refers to a single destination host
- *Broadcast* (one to all): refers to every host on a network or subnet
- *Multicast* (one to many): refers to a group of hosts in a network, not necessarily all of them.

At the very beginning of IP Addressing, there was not the concept of variable length netmask, but a classful concept. Now it is not used anymore.
This means that there are classes of IP Addresses with a predefined fixed netmask, so it can be determined directly by looking at the IP Address.
There are 5 different classes of IPv4 Addresses.

Five Different Classes of IPv4 Addresses

Class	First Octet decimal (range)	First Octet binary (range)	IP range	Subnet Mask	Hosts per Network ID	# of networks
Class A	0 – 127	0XXXXXXX	0.0.0.0-127.255.255.255	255.0.0.0	$2^{24}-2$	2^{7}
Class B	128 – 191	10XXXXXX	128.0.0.0-191.255.255.255	255.255.0.0	$2^{16}-2$	2^{14}
Class C	192 – 223	110XXXXX	192.0.0.0-223.255.255.255	255.255.255.0	$2^{8}-2$	2^{21}
Class D (Multicast)	224 – 239	1110XXXX	224.0.0.0-239.255.255.255			
Class E (Experimental)	240 – 255	1111XXXX	240.0.0.0-255.255.255.255			

Figure 4: IPv4 Classes

Class A has 8 bits for the network (the first bit is fixed to 0) part and 24 bits for hosts: so there are 128 networks with each one 224 hosts.
Class B has 16 bits for the network part (the first two bits fixed to 10) and 16 bits for hosts: so there are 214 networks with 216 (65536) hosts each one.
Class C has 24 bits for the network part (the first three bits fixed to 110) and 8 bits for hosts: so there are 221 networks with 28 (256) hosts each one.
Class D is reserved for multicast address, Class E is experimental.
If there is a 0 in the first bit: it is a class A. If there is a 10: it is a class B. If there is a 110: it is a class C. If there is a 1110: it is a class D.

It is important to notice that there are routable and non-routable address ranges.
The routable addresses need to be unique on the Internet.
The non-routable address ranges are defined in *RFC1918*[2].
Of the approximately four billion addresses defined in IPv4, about 18 million addresses in three ranges are reserved for use in private networks.
Packets addresses in these ranges are not routable in the public Internet; they are ignored by all public routers.
Therefore, private hosts cannot directly communicate with public networks, but require *NAT Network Address Translation* at a routing gateway for this purpose.

RFC 1918 name	IP address range	Number of addresses	Largest CIDR block (subnet mask)
24-bit block	10.0.0.0 – 10.255.255.255	16 777 216	10.0.0.0/8 (255.0.0.0)
20-bit block	172.16.0.0 – 172.31.255.255	1 048 576	172.16.0.0/12 (255.240.0.0)
16-bit block	192.168.0.0 – 192.168.255.255	65 536	192.168.0.0/16 (255.255.0.0)

Figure 5: Private IPv4 Addresses Range

Let's make some examples of IP Addressing.
Remember that two hosts in the network must be used for the network address and the broadcast address.
The network address is the one with the host part composed by all bits put to 0, the broadcast address is the one with the host part composed by all bits put to 1.

[2] *Private Address Space, RFC 1918*: https://datatracker.ietf.org/doc/html/rfc1918

IP Addressing Examples:

192.168.5.85/24
IP Address: 192.168.5.85
Subnet Mask: 255.255.255.0 (/24)

IP Address: 11000000.10101000.00000101.01010101
Subnet Ma: 11111111.11111111.11111111.00000000
AND Oper: 11000000.10101000.00000101.00000000

The result of the AND is 192.168.5.0.
There network address is 192.168.5.0 and the broadcast address is 192.168.5.255.
There are 232-24 – 2 = 28 – 2 = 256 – 2 = 254 hosts from 192.168.5.1 to 192.168.5.254.

	Hosts	Netmask	Amount of a Class C
/30	4	255.255.255.252	1/64
/29	8	255.255.255.248	1/32
/28	16	255.255.255.240	1/16
/27	32	255.255.255.224	1/8
/26	64	255.255.255.192	1/4
/25	128	255.255.255.128	1/2
/24	256	255.255.255.0	1
/23	512	255.255.254.0	2
/22	1024	255.255.252.0	4
/21	2048	255.255.248.0	8
/20	4096	255.255.240.0	16
/19	8192	255.255.224.0	32
/18	16384	255.255.192.0	64
/17	32768	255.255.128.0	128
/16	65536	255.255.0.0	256

Figure 6: Netmask and number of network bits

10.128.240.50/30
IP Address: 10.128.240.50
Subnet Mask: 255.255.255.252

IP Address: 00001010.10000000.11110000.00110010
Subnet Ma: 11111111.11111111.11111111.11111100
AND Oper: 00001010.10000000.11110000.00110000

The result of AND is 10.128.240.48.
The network address is 10.128.240.48 and the broadcast address is 10.128.240.51. There are 232-30 – 2 = 22 – 2 = 4 – 2 = 2 hosts from 10.128.240.49 to 10.128.240.50.

A note about IP Address is that in a *point-to-point link*, so a communication between two single hosts, using a 30-bit netmask is waste because for example if the host A sends a broadcast only B will receive it.
So, there is a proposal on *RFC 3021* for use a 31-bit prefix on IPv4 point-to-point links in order to reduce the waste of IP Addresses in a subnet.
To reduce the waste of IPv4 addresses the *temporary* solution is the NAT, and the permanent solution must be the use of IPv6 addresses.

II. Network traffic monitoring

- Layering concepts

The idea is that we try to see what happens in the network.
When there is a network realized between hosts, it is realizing the movement of packets. A packet is a sequence of bits.
To make the device able to recognize what is the packet, when it starts, when it ends, and so on there must be an agreement on the concept of standard and a set of common rules defined by a *protocol*.
A protocol is made for providing some characteristics and to make the device able to understand the data it is receiving.
The network must be reliable and flexible, technology and its implementation could change, but this must be independent from the working of the network.
This is realized using the concepts of *layers*.
We arrange the whole ideal communication between two hosts trying to abstract the single real steps needed to realize the communication.
Every step is done and realized by a specific layer.
So, the message that we want to send, will be processed step-by-step, by several layers and changing the structure of the data.
The communication between hosts in the networks is organized in tasks, each assigned to a layer.
Every layer offers a service (a host of facilities) to the users in the layer above and exploits the services offered the layer below.

- Encapsulation/decapsulation and layered architectures

The data that are supposed to be transferred from the application layer to application layer, is in some way, transferred and transformed by several layers so that can be moved in the network.
Every layer adds some protocol information and provides data to the layer below, this is called *envelope*.
Then, at the very bottom of the “stack layer”, the physical layer sends data over physical medium to the destination.
Data are at this point transformed in physical forces, like radios waves, lights, electromagnetic waves; so, some measurable forces.
Then, when these data are received at the destination, the data will be transformed again from a physical force to digital values and will be send up to the stack.
Each protocol in the destinations reads the appropriate protocol information and forwards the data to the layer above.

The encapsulation is done, by including new information that are dependent from the next layer, when moving from one to the lower layer.
When the data arrives to the destination, then there is a decapsulation, from the bottom layer to the upper layer.
Remember that when the data arrives to the destination, this maybe is not the real destination but only an intermediate destination: this happens always with routers and switches.
The idea is that packets moving in a physical medium *usually are destroyed and recreated*; it is a copy of the original data that arrives at the destination.
When moving data from an ethernet network to another, by crossing a router, the original frame is destroyed and a new frame is generated with a different address, for example there is a switch in the MAC Address of the destination.
When the devices are in the same network, they are directly connected, and it is possible to send packets directly using their MAC Addresses.
If the devices are not in the same network: the gateway receives the packet, it will destroy the frame and create a new frame with its gateway MAC Address as source and the next-hop MAC Address towards the destination.

FIGURE I.30 The ISO protocol stack.

Figure 7: The ISO model

The ISO/OSI model is based on a reference model with 7 layers, and TCP/IP model is based on reference model with 4 layers.
The common idea is that there is a ***packet switched network***: every packet is independent from all the others, it should contain inside all the details to reach the destination.

In the OSI model, the upper 3 layers are application, presentation and session.
Then the Transport Layer is very similarly related to the session concept, and it is responsible for the delivery of the data, to distinguish several instances running on the sender/destination hosts, for rearrange the data.
To this layer belongs the *TCP* and *UDP* protocols.
Then there is the most important layer of Internet that is the *Network Layer* that realizes the routing mechanism, thanks to the IP protocol.
To this layer belongs the *ICMP* protocol, that is used to check the status of network.
Then there is the *Data Link layer* (PPP, Ethernet protocols) and the Physical layer that is responsible of changing the data in measurable physic forces to transmit over the media.

In the *TCP/IP* Model the Application, Presentation and Session layer are merged in a unique single ***Application Layer*** (*SMTP, HTTP, FTP*, …) .
Transport Layer (*TCP, UDP*) is the same, and the Network Layer is called ***Internet Layer*** (*IP, ICMP, IPSec*).
The ***Network Access Layer*** groups the Data Link Layer (Ethernet, Wi-Fi, ARP) and the Physical Layer.

- Client-server communication

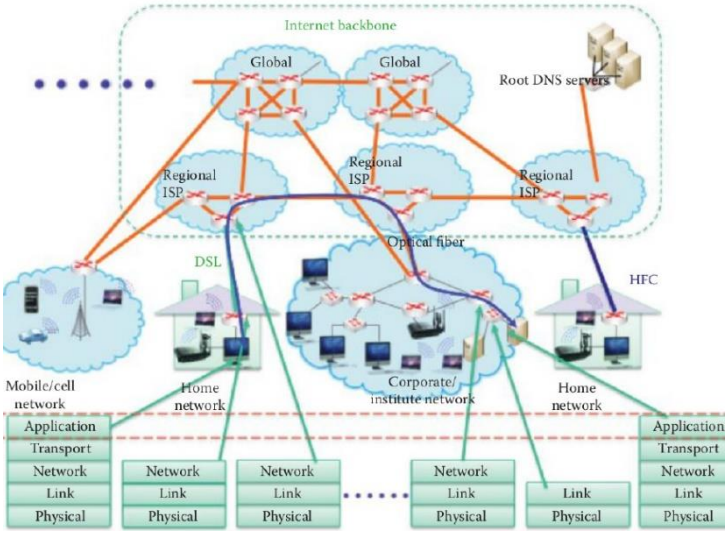


Figure 8: Client - Server communication

This is an example of client-server communication.
We imagine the process of communication from host to reach the server.
The host generates some data; the final purpose of the data exchange is that the application layer should directly communicate with the application layer running on the destination server.
So, remember that in this layer mechanism there a 1-to-1 map between the “stack layers” of the source and the destination.
For example, the application layer of source is supposed to communicate to the application layer of the destination host.

These below are all the steps of the client-server communication.
Inside the packet there are information related to the Transport Layer at the destination that are strictly bound to the application on the server.
In the host, the Application layer, insert into the data and tells the Transport Layer the destination address of the server.
For example, my source host generates HTTP request for the application running on the server-side on port 80.

We include the IP Address of the destination; this is important also for all the intermediate hops before reaching the final destination.
Because the destination IP Address is what makes possible to route the packet: when a router receives the packet, looking at the destination address, can decide where moves it by looking the routing table.
Once the Network Layer will include the destination IP Address, then there is the Link Layer.
For the data link, when the host must decide the next-hop for the destination, it must look at the IP Address of the destination and its Network Mask by comparing them. Then, it will see that the two network are different and the IP Address is not directly reachable.
So, in this case, the destination MAC Address that will be inserted in the packet is *the one of the gateway of the network of the host*.
In the Link layer will be inserted as source MAC address the one of the host that generated the data, and destination MAC address the gateway of the network.
In the Physical Layer, the data are transformed depending on the medium used (WiFi, fiber, etc..).

What happens when data are received by the gateway.
The Physical Layer of the gateway will receive the packet and it will see that it is the destination of that packet because of its MAC Address in the destination.
But it is not the final destination, because the IP Address destination is not it's one but the one of the external server.
It has to realize who will be the next-hop: it will look at the server IP Address and its netmask, realizes that it is a remote IP of a remote network.
The gateway has choose between the different links to which it is connected.

A *new packet* is generated by the router with source MAC Address *the gateway MAC Address* and the destination *MAC Address of the next-hop*.
This destroy and recreation of packet happens every hope of the path; packet is destroyed and regenerated in another link.
Also, by moving up and down through the layers, the data will change and modified the envelope by encapsulate and decapsulate every time.

The *unique exception* is when the packet arrives at a *switch*: the packet is received and simply is regenerated and forwarded only in the segment of the network where there is the final destination, *without changing* anything in the source MAC or destination MAC or other property.

The destination router will know that the IP Address belongs to the same network and so it's a local destination.
The server will receive the packet and reads it is the final destination in the destination MAC Address of the link layer; the destination IP Address in the Network layer it is also its own IP Address.
It will remove the envelope and retrieve information for the transport layer to understand which is the specific thread of the many threads that are waiting for data and finally transfer the data to the final application that was waiting for these data.

So basically, every layer the packet is encapsulated and decapsulated, by moving it up and down through the layers.
Every step, every next hop (except the switch) some fields of the packet are changed like the source and destination MAC Address; so the packet is destroyed and regenerated as a new one.
IP address destination usually does not change, instead MAC Address changes every hop.
The only thing that does not change in this chain it's the payload related to the application and possibly something related to the Transport Layer.

With the *Network Address Translation NAT*, one single IP Address that masks several others IP Addresses, there could be some changes on the IP Source Address or Destination in the packet.
The Transport layer has the illusion of a direct end-to-end connection between processes in arbitrary systems, it does not realize what happens in the between of the network.
The Network Layer is transferring data between arbitrary nodes; it knows that there are several nodes in the network to which data must move.
Data Link is blind, only transferring data between directly connected systems via direct cable, only knows a local link, only know a small fragment of the entire path.

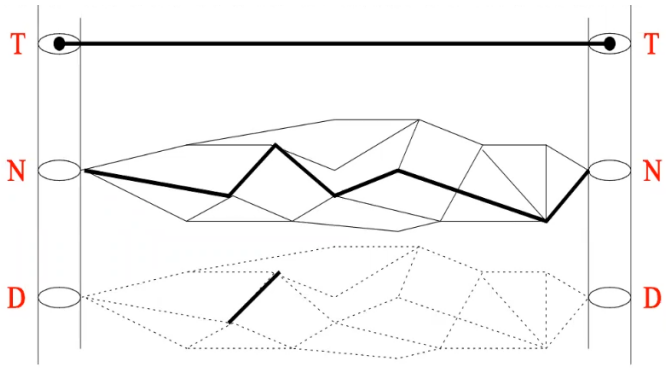


Figure 9: Layer Ideal Representation

- ## Addresses in the architectures

Every layer has different protocols and an own address.
We've mentioned the Data Link Layer has the *MAC Address* and the Network Layer has the *IP Address*.
For the Application Layer there are the *Internet names* or the *host names* (www.sapienza.it) and for Transport Layer there are *port numbers* in the range [0 - 65535] that allow to identify different applications running on the same host and also to determine the type of service we want to access.

Ports
The ports are in a range of [0 - 65535].
The **source port** is randomly chosen by the OS, usually on a number in a range [49152 - 65535] and are called *ephemeral ports*, so used only for a single session; it is used to distinguish different requests of different applications or of the same application.
The **destination port** determines the required service (application):
- there are assigned ports [0 – 1023] called *well-known ports* and used by servers for standard internet applications: 21 FTP, 22 SSH, 25 SMTP, 80 HTTP, 143 IMAP, 443 HTTPS and so on.
- then [1024 – 49151] the register ports can be registered with *Internet Application Naming Authority* (IANA).

- ## IPv4 Header

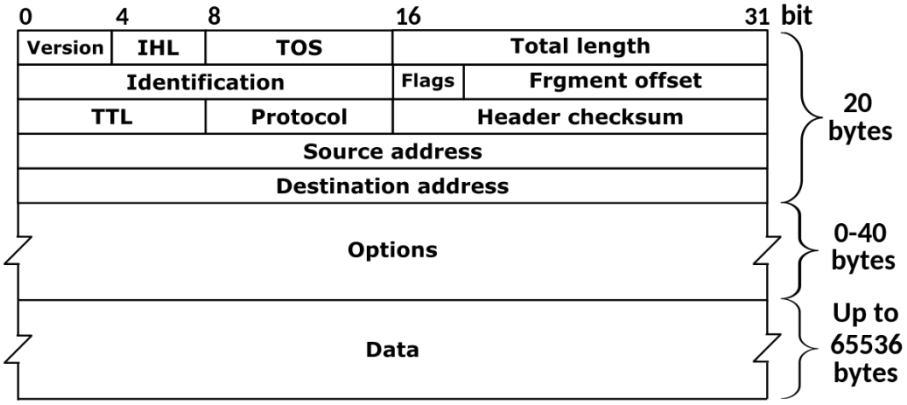


Figure 10: IPv4 Header

An IPv4 packet is composed by a fixed header that has a size of 20 bytes.
Then there is the *Options Header* that could be from 0 to 40 bytes.
Therefore, an IPv4 header could have a size from a minimum of 20 bytes to a maximum of 60 bytes (excluding the data payload).

IHL (Internet Header Length) specifies the size of the current header.
TOS (Type Of Service) usually is not very used, is ignored, and it has a size of 1 Byte. *Total Length* specifies the total size of the IP packet, header plus data, and it has a size of 2 Bytes.

Identification (2 bytes), *Flags* and *Fragment Offset* (both occupies 2 Bytes) headers are used if a packet needs to be fragmented depending on the *MTU* (*Maximum Transmission Unit*) of a link.
Identification is used to indicated that a packet belongs to a specific stream of packets; Flags and Fragment offset are used to indicate the order of the fragment or if there are other fragments to come, or it is the last one.
Fragmentation is one of sources of weaknesses of IP; IPv6 includes fragmentation mechanisms and in the next evolution of the standard and they decided as much as possible to reduce fragmentation.

TTL (Time To Live) is used to avoid a packet remain alive forever in the network without reaching a destination; every time the packet is forwarded, the TTL is decreased by one, and when eventually TTL reaches 0 the packet is removed.
At this point, when TTL reaches 0, there is a ICMP packet with “Destination unreachable, TTL expired”. TTL is used to realize traceroute.
This can happen if there is a broken routing table, a misconfiguration, so there is an infinite loop.

Traceroute is realized in this way: the first packet has TTL=1 when it reaches the first hop router A, TTL = 0 and router A sends an ICMP, this is the first hope; increasing by one, we have TTL=2 and when it reaches the second hope, router B has TTL=0, and sends an ICMP, this is the second hope and so on.

Protocol field is used for the protocol associated to the payload (TCP, UDP, etc..), *Header Checksum* for check mistakes in the header, and then obviously *Source Address* and *Destination Address*.

- ## TCP and UDP

TCP and UDP are the different transport protocol and are used on the transport layer; both of them relays on the IP protocol.
Usually we have that an application protocol uses only one transport protocol but there are several exceptions like DNS service that can use both TCP and UDP. Also OpenVPN can use both TCP and UDP.

Connection-less
The UDP protocol does not make any control on the data exchange, does not have any flow control and no reliability. Also packets have no sequence numbers. When a packet is received, there is no possibility to recover a miss packet or something like that.

Connection
In TCP protocol there is a reliable data exchange, flow control and reliability. You can regulate the amount of exchanging data, you can acknowledge the packets that are lost in the network and receive again the missing ones, and there's also a way to rearrange packets in the correct order.

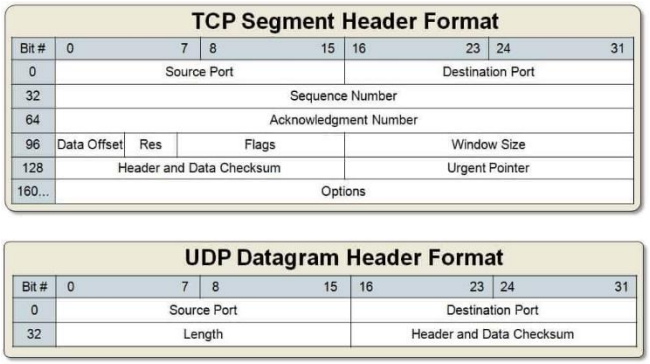


Figure 11: TCP and UDP headers

Obviously these differences have a cost in terms of header complexity for the two protocols.
Source Port and Destination Port number have the same size (2 bytes each one).
The UDP Header is extremely simple: ports, length, checksum and data.

The TCP header has a cost for reliability and flow control: sequence number field, acknowledgment number, window size, bits for establishing connection, TCP checksum field.
UDP has a simpler and shorter 8-byte header compared to TCP's default header size (at least) of 20 bytes.

TCP connection

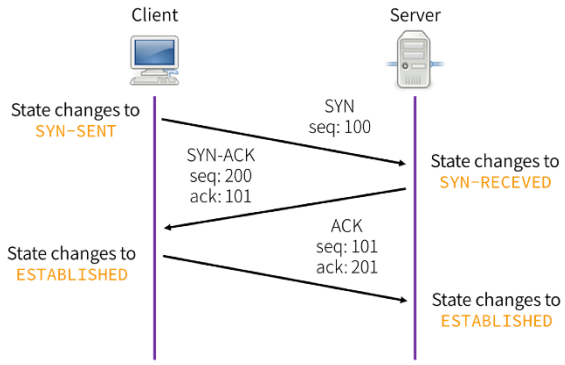


Figure 12: TCP handshake connection

The TCP connection is called 3-ways handshake. This because there are 3 packets involved in the connection establishment.

1) When you want to open a new connection, the client sends a **SYN packet**: it is a packet with no data, with the SYN flag on and with sequence number that is a random value x.
The random value is needed to make harder hijack the TCP connection for an attacker.

2) The receiver, after the SYN packet is received, sends a **SYN-ACK packet** to the sender.
It sets its own sequence number that is a number y and the ACK number is the previous random sequence number x+1.

3) The client receives the SYN-ACL packet, change its state to ESTABLISHED (ready to communicate) and sends to the server a **ACK packet** with sequence number equal to x+1 and ack equal to y+1.

4) The server receives the ack and change its state to ESTABLISHED.

In the past, there was an implementation of TCP, in which there were some weaknesses in the random number generator and it was proven to be extremely easy to be hijacked.
Now it is no more like that.

Services in TCP/UDP

Some of services relying on TCP: FTP on port 20/21, SSH on port 22, Telnet on port 23, SMTP on port 25, HTTP on port 80, IMAP on port 143, SSL on port 443.

Some of services relying on UDP: DNS on port 53, DHCP on port 67/68, TFTP on port 69, SNMP on port 161, RIP on port 520.

- ## DNS (Domain Name Server)

The *Domain Name System (DNS)* is the hierarchical and decentralized naming system used to identify computers reachable through the Internet or other Internet Protocol (IP) networks.
These are most used to map human-friendly domain names to the numerical IP addresses computers need to locate services and devices using the underlying network protocols but have been extended over time to perform many other functions as well.

So, a DNS is service to get the IP address from a human friendly domain name, like www.sapienza.it.
There is a hierarchical way to distribute the DNS names, the idea is that the names are partitioned according to their structure.

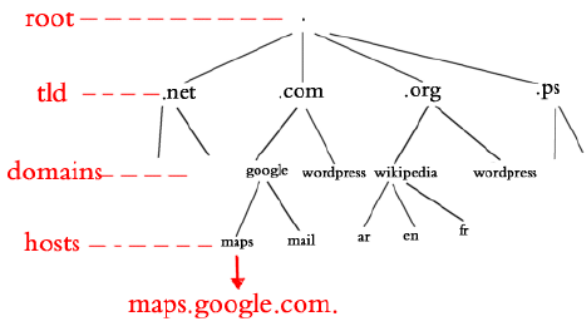


Figure 13: Tree structure of the domain name space

The domain name space consists of a tree data structure.
Each node or leaf in the tree has a label and zero or more resource records (RR), which hold information associated with the domain name.
The domain name itself consists of the label, concatenated with the name of its parent node on the right, separated by a dot, this means that the mechanisms that are supposed to be used are recursive.

The DNS query is at the very end a recursive call: the same action is applied several times up to resolving the IP Address.

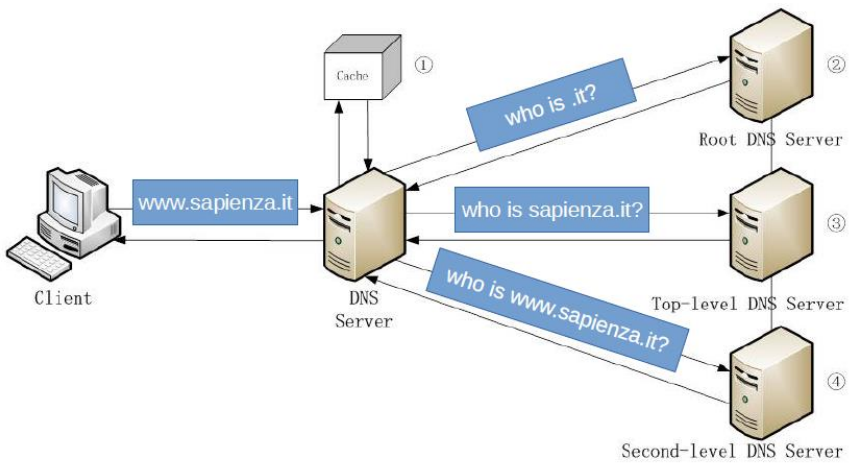


Figure 14: Recursive DNS query

The first step for the client is to check its local check if it has already asked for resolving the DNS name, if yes then it is not needed to ask to a DNS server to resolve it again.
If not instead, the client will send a *recursive DNS query* to the DNS server for the IP Address, for example, of www.sapienza.it.
The DNS uses a recursive mechanism starts from .it, and asks to the *Root DNS Server,* which is the DNS that knows all the names ending with .it.
The *Root DNS Server* will give this IP Address of a *Top Level DNS Server*, then the DNS server will ask for the sapienza.it IP address to the *Top Level DNS* server and so on until the entire string is reconstructed.
The DNS Server usually also has a *DNS cache,* a space of memory, that is used to store the resolved IP Addresses for some time until a special time of expiration (in order to avoid asking for the resolution every time).

- ## DHCP (Dynamic Host Configuration Protocol)

It is a client-server mechanism.
The server has a pool of IP addresses to distribute, together with the network configuration.
The client requesting a new IP address receive a proposal and accept it.
Once accepted, the IP is reserved for a "*leasing time*".

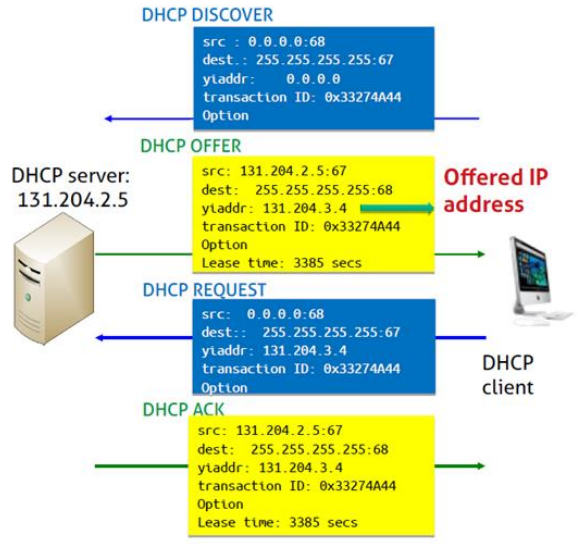


Figure 15: DHCP procedure

These are the steps of a DHCP procedure:

1) Host broadcasts a "DHCP Discover":
The strange is that host sends a packet without an IP address asking for an IP address, and the destination is *unknown* (the host does not know the address of the DHCP server), it is like sending a message in a bottle in the sea.
The source address is 0.0.0.0 and the destination IP address is 255.255.255.255.

2) DHCP server responds with "DHCP Offer":
This message has a source IP address (the one of the DHCP server) and the destination is 255.255.255.255 with the proposal IP address *yiaddr* and some options, like the domain name, the lease time, and so on.

3) Host request IP Address with the "DHCP Request":
The host (with source IP 0.0.0.0) specifies with this message that it is requesting the previous proposal IP Address.

4) DHCP server sends the "DHCP Ack":
DHCP server accepts the agreement and specifies the leasing time.

- Capturing packets using Wireshark

We capture the packets flow in the network, using a network traffic dump tool.
The *dumpcap* is the less advanced version of the capture.
The *tcpdump* can also use some filters.
Wireshark and tshark are more advanced because they can also visualize the capture packets and can also analyze and decode the single elements of the packets. All of them are based on *pcap* library.
We will use *Wireshark*, because is the more advanced and has a graphical interface to visualize the packets.
tcpdump also will be used because is the only one that is possible to use on servers (that don't have a graphical interface) it is possible to use it on command line and is the best tool to perform ***diagnostic***.

Wireshark
Data from a network interface with Wireshark are dissected by recognizing the different elements of the data.
The dissection is in:

- *frames (level-2, ethernet)*
- *packets (level-3, IP)*
- *segments (level-4, tcp/udp)*

Then they are interpreted and visualized in the context of the recognized protocol.
We usually use for capturing the packets the ***promiscuous mode*** (or *monitor mode*) of the interface card.
The card (data-link layer) only forwards to the upper layers packets that are intended for the host.

We've said that potentially, in the Ethernet network, every host could receive every packets, also not intended for the host.
But we said also that this is avoided by looking at the destination MAC Address: if the destination MAC address is the MAC of the host, or it's a broadcast MAC address or it's a multicast MAC address in which is included the MAC of the machine, only then in this case is forwarded to the upper layer (the Network Layer).

This is obviously a limitation if you want to check the data that are passing on all the network, you can't perform this kind of check, because all the packets have to be delivered to Wireshark.
So, the promiscuous mode bypasses this limitation: all the data received will be delivered to the upper layer, without performing any kind of filtering.
This mode is called "*Monitor Mode*" when used in Wi-Fi network.
The difference is that in monitor mode a Wi-Fi network card is supposed to capture every packet even if it belongs to a different SSID, different network you are connected on.

These are the reasons when to use Wireshark:

- looking for the root cause of a known problem (so if we already know that there is a problem, not to find it).
- look for a given protocol or stream between devices.
- analyzing specific timing, flags, or bits on the wire;
- following a conversation between devices (this is not possible with tcpdump).

So, it should not be the first tool for discovering a problem but is the one that must be used when a problem is already known to try solving it.

Frames are collected from the interface and passed to several, consecutive, "dissectors", one for each layer.
It means that a sequence of bits 0 and 1, and try to apply some forms of understanding for these sequences.
Dissectors means that there is a process from the outer to the inner envelope, trying to understand, according to given protocols for every layer: for example, Ethernet, then IP protocol, then UDP protocol and then inside DNS.
So the idea is to have several dissectors, everyone specialized on one single protocol. Frames pass from bottom layer to upper layer.

Protocol can be detected in two ways:

- *directly*, if the protocol has the field that states which protocol it is encapsulating.
  This is common: Ethernet tells the protocols it is carrying on, IP also does it, TCP/UDP makes it easier to guess the protocol by looking at the port number.
- *indirectly*, with tables of protocol/port combinations and heuristics; usually it works, but there are some troubles when protocols are used in non-standard ports.

Alternative way to capture traffic are *Netflow* and *Zeek* protocols mechanisms. Netflow is the standard for switches and routers to collect statistics on the traffic; Zeek is a framework used for traffic inspection and as an intrusion detection system.

Wireshark Filterings

There are two kinds of filters: *display filters* and *capture filters*.

- **Capture filters** will specify only the kind of packets that are going to be processed; so, the packets that will not match the filter will not be captured and lost forever. This is a physical limitation, packets that does not match the filter will not be processed.
- **Display filters** to inspect only the packets you want to analyze without losing any packets, so once the data has been processed.

Capture filters uses the *BPF Berkeley Capture Filter* syntax that is:

- *protocol direction type*

For protocol we can have, for example: tcp, udp, ether, ip, ip6, arp
For type we can have: host, port, port-range, net
For direction we can have: (omitted), src, dst
Other primitives: less, greater, gateway, broadcast
Combinations with operators: and, or, not

Display filters, display only captured packets matching the filters, so the packets are not discarded or lost. It is easy but refined syntax: only packets evaluating true are displayed. It is possible to use comparison operators, filters use types, and common logical operators.
It is interesting that is possible to build filters by interacting with packets (for example, clicking on a packet and set filter packets like this or with these fields).

Promiscuous mode limitation

The limitation with the promiscuous mode is that it can work only with the hub, because with the switch every host as its own port to which is directed connected, the packet will be delivered only to the real destination.
In a segmented network, that is the one build by the switch, the hosts are not in the same segment; we will see the packet only if hosts are in the same segment and this is not the case.

Then the solutions could be:

- *physical tap*, like a splitter.
- the *port mirroring* on a managed switch: in the switch, one special port will be used to receive all the packets that are moving through the switch.

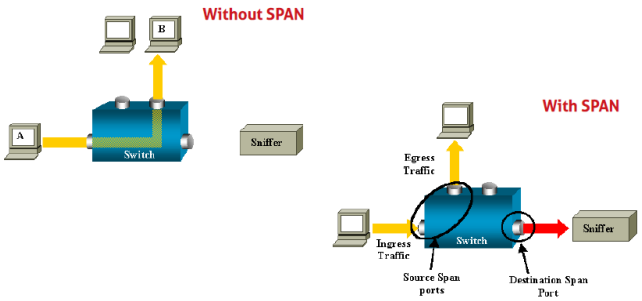


Figure 16: SPAN port in a switch

The port mirroring, like in the CISCO naming, is called *SPAN (Switched Port Analyzer*) or *Roving Analysis Port (RAP)*: the traffic is replicated on the SPAN port.

- there are also more aggressive approaches: *ARP cache poisoning, MAC flooding, DHCP redirection, redirection* and *interception with ICMP*.
- on virtualized environments and SDN, this can be easier (Kathara environment, it is possible to capture everything) or harder.

How to prevent packet capture:

- *Dynamic address inspection:* it is hardware Implemented in switches, the Dynamic Address Resolution Inspection (DAI) validates ARP packets. IP-to-MAC address binding inspection, drop invalid packets
- *DHCP snooping:* it is hardware implemented in switches, it distinguishes between trusted and untrusted ports and uses a database of IP-to-MAC. Ports that show rogue activity can also be automatically placed in a disabled state.

It is possible to filter, instead of using hostnames or IP addresses, you could also use country or city names by configuring GeoIP resolver, for example to capture traffic that comes from China:

- ip.geoip.country eq “China”

III. IPv6

- Introduction to IPv6

IPv6 is not really a new protocol.
It was started to be developed in the mid of 1990s and it was designed to be the successor of IPv4.
The change from IPv4 to IPv6 primarily into the following categories: expanded addressing capabilities, header format simplification, flow labeling capability, authentication and privacy capabilities, support for extensions and options.
The most important novelty of IPv6 is the addressing size; the address space is 128-bit, written in hexadecimal.
This means that there could be 2128 different IPv6 addresses: 340 undecillions!
In this way it is possible to give an address to each atom on the Earth.
With the evolution of the Internet, every device can connect to the Internet, think to the IoT (Internet of Things), and in this way it is not possible to have a saturation of the IPv6 address space.

IPv6 is not just about more addressing space:

- Stateless autoconfigurations is something similar to this reasoning: there are a very big number of IPv6 addresses, so get the one that you choose.
  You only need the network address.
- End-to-end reachability without private addresses and NAT: we do not need anymore the NAT mechanism because of the huge quantity of IPv6 addresses; same for the private addresses.
- Better support for mobility.
- Peer-to-peer networking easier to create and maintain.
- Services like VoIP and QoS become more robust.

IPv6 started in 1993 with RFC 1550: IP, Next Generation.
IETF changed later the address size from 64 bits to 128 bits.
The very first mention of IPv6 was in 1995. There was also a project related to IPv5.
The need of IPv6 was because of the IPv4 address space saturation.
In 2011 IANA allocated the last /8 IPv4 addresses block.

The NAT has been used in IPv4 also to help “hide” customers and works for many client-initiated applications. However, NAT also create some issues, like peer-to-peer networking and accessing our “hidden” systems from other networks.
Using NAT to “hide” IPv6 networks has been source of some debate; however, IETF continues to state that NAT is not intended as a security feature.

- IPv6 Address Representation

Hex	Binary	Hex	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Figure 17: Binary to Hex

IPv6 use the hexadecimal notation to represent the IPv6 addresses.
This because 1 hex digit represent 4 bits.
Since IPv6 addresses are composed by 128 bits, we need 32 hexadecimal digits to represent an IPv6 address.
These 32 hexadecimal digits are divided into 8 groups of 4 hexadecimal digits, that means 8 groups of 16 bits.
Each group is separated from the other by a colon ":".
Every group of 4 hexadecimal could go from 0000 to FFFF.

Remember that every digit in the IPv6 address is an hexadecimal character, so is 4 bits, so the first group 2001 for example is actually in bit: 0010 0000 0000 0001.
Just for a comparison, an IPv4 was made by only two groups of 4 hexadecimals.

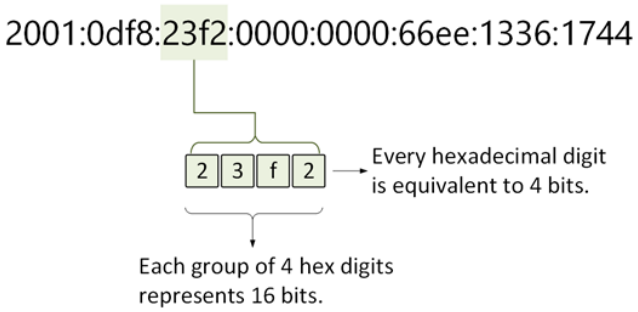


Figure 18: IPv6 Address Hexadecimal Representation

Rules for compressing IPv6 Addresses

1) Omitting the Leading 0s: leading 0s in any of the 4-hexadecimals group do not have to be written. Trailing 0s must be included instead.

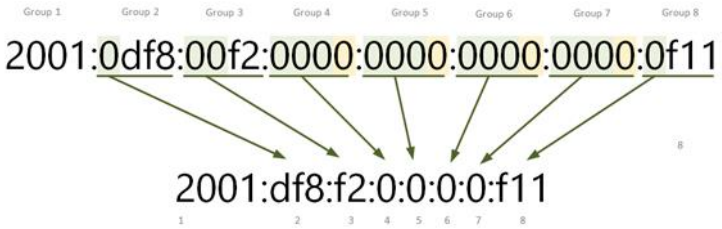


Figure 19: IPv6 Address Compression, omitting leading zeros

2) Double colon :: Any single, continuous string of one or more 4-hexadecimals group consisting of all zeroes can be represented with a double colon (::).

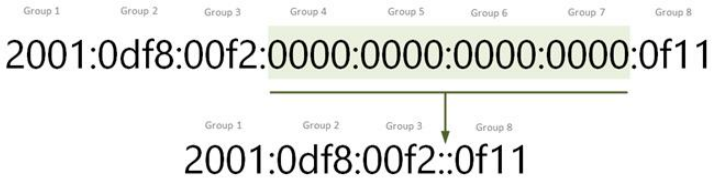


Figure 20: IPv6 Address Compression, double colon

If there are multiple possible reductions, the longest string of zeroes must be replaced with (::); if they are equal only the first strings of 0s should use the :: representation.
Combining the two rules, the previous IPv6 address will look as **2001:df8:f2::f11**.

Note that the IPv6 address that begins with **2001:DB8**, it is a dedicate IPv6 address used for documentation; it is not possible to see this address in the public Internet.

- IPv6 Addresses Types

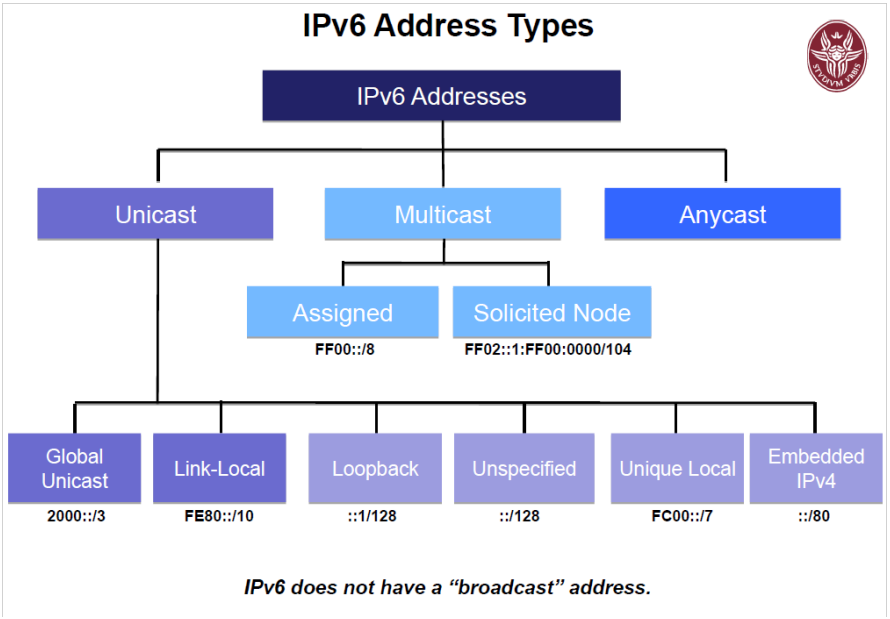


Figure 21: Types of IPv6 Addresses

The IPv6 Global Unicast Address is the equivale of IPv4 public address.
IPv6 does not have a "broadcast" address.
IPv6 Source Address is always a Unicast Address, that could be Global Unicast or Link-Local Unicast.
IPv6 Destination Address could be instead Unicast, Multicast or Anycast.

Global Unicast Address

The Global Unicast Address GUA goes from **2000::/3** to **3FFF::/3**.
The range is from **2000::/64** thru **3FFF:FFFF:FFFF:FFFF::/64**.
In a binary representation we have that 2000::/3 is: 0010 0000 0000 0000 … 0000, with ***the first 3 bits are fixed*** by the netmask for the prefix of the address.
The first group fixed is 001x with the bit x that could be change to 1, so we have that the first group could be also 0011 that is 3.
Then all the others 0000 blocks could be 1111 that in hexadecimal is F.
So the maximum we can have is 3FFF for the first group.
The GUA are globally unique and routable and are the analogue of the IPv4 public addresses. The network **2001:DB8::/32** is reserved for documentation.

The range used for the Global Unicast is using only 1/8 of the full addressing space; so for the whole global addressing only a fraction of the entire pool is used.
The other fractions are not allocated now except some small fractions in 111.

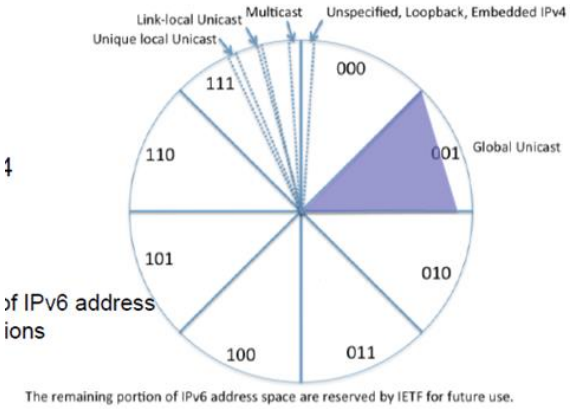


Figure 22: Division of the Full IPv6 Addressing

All end users will have a GUA.
Note only but, an interface of a host potentially/usually has multiple IPv6 addresses on the same or different networks.
This fact provides some features like the protection of the topology, like it is possible to use an IPv6 address only for a limited amount of time, and then change it. This feature is called *Privacy or Temporary Address*.

The **Prefix** is equivalent to the network address of the IPv4 address.
The **Prefix Length** is equivalent to the subnet mask in IPv4.
Interface ID is the equivalent to the host portion of an IPv4 address.

The GUA is divided into 3 parts: *Global Routing Prefix*, *Subnet ID*, and *Interface ID*. Differently from IPv4, when we must keep the network size, in IPv6 since there are so many addresses, usually *we assume that the last 64 bits are specific for the* **Interface ID**, so 64 bits are used only to represent the hosts, the equivalent of 2^{64} hosts, 18 quintillion. The last 4 blocks of hexadecimals are used for the hosts.

Usually, discussing about GUA, we have that of the first 64 bits:

- the **Global Routing Prefix** is composed by 48 bits, /48, so the first 3 blocks of hexadecimals.
- the **Subnet ID** is composed by 16 bits, so the 4th block of hexadecimals. With 16 bits we can represent 65536 different subnets.

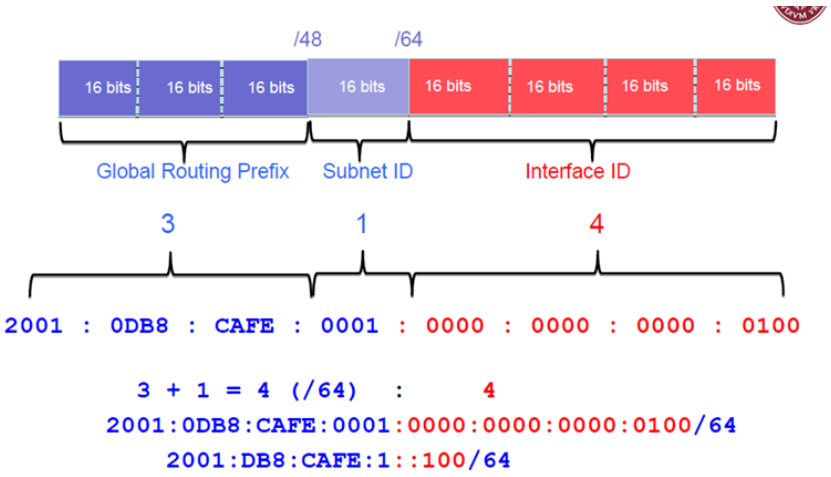


Figure 23: GUA division

The ISP will give usually to the user a Global Routing Prefix of /48 and then the user is free to decide how to subnet that prefix up to 65536 subnets (by using the 4th block).
The **3-1-4 Rule** says that: the first 3 blocks are related to the Global Routing Prefix, the one following is related to the Subnet ID and the remaining 4 are used for the interface id.

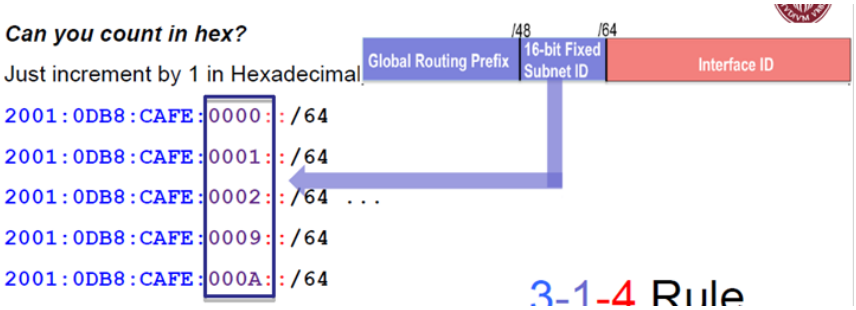


Figure 24: 3-1-4 Rule IPv6 Address

Link-Local Unicast

The Link-Local Unicast addresses are **NOT** routable off the network.
They are intended to be used only locally, they're valid in one single link, local to that link or network.
They are extremely useful and used in IPv6 for the stateless autoconfiguration: they are used to properly configure the network.
An IPv6 Link Local Unicast **must be unique on the link**.

The Link-Local Unicast **is not included in the IPv6 routing table**.
An IPv6 device **must have at least** a Link-Local Unicast IPv6 Address.
They are even less then private addresses in IPv4: with Link Local it's used

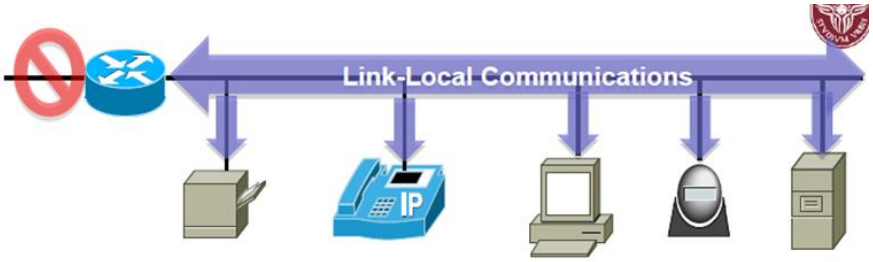


Figure 25: Link-Local Communication

Every IPv6 Link-Local Unicast address is in the range **FE80::/10**.
Going to the binary representation of **FE80::/10**: **1111 1110 0100 0000 … 0000**
The first two blocks **1111 1110** represents the hexadecimals digits **FE** and they are fixed because they represent the first 8 bits.
Because we have the first 10 bits that are fixed, the first 2 bits of the 3rd block are also fixed: **10**xx, only the 9th and 10th bit are fixed.
This means that the third hexadecimal could go from **8** (that is 1000) to **9**, **A** and **B** **(**that is 1011**)**.
So the range for a Link Local Unicast is from:
FE80::/10 to **FEBF:FFFF:FFFF:FFFF:…/10**.
Usually a Link-Local Unicast Address has the first 64 bits equal to FE80:0000:0000:0000 but the range as seen is up to FEBF:FFFF:FFFF:FFFF.

The Interface ID is composed by the last 64 bits of the Link Local Unicast Address.
The Interface ID is generated usually using an algorithm that uses the MAC Address of the device, because we know that it is almost impossible to have two devices with the same MAC Address.
The algorithm that generates the interface id is called *EUI-64 (Cisco routers)*.
It is possible also that:

- the host operating system generates randomly the 64 bits used for the Interface ID.
- it is possible to do a static manual configuration by set manually the interface ID, that is common practice for routers to have a Link Local Unicast like **FE80::1/10**.

EUI-64 Algorithm for Generating ID Interface

The idea is to define a deterministic procedure that is able to obtain an interface's identifier derived directly from that interface's link-layer address MAC Address (Layer 2 Address) by using these steps:

- MAC Address is 48 bits long string. Split the MAC Address in two strings of 24 bits long.
- Insert the hexadecimal string **FFFE** (16 bits) in between the two obtained parts. We got at this point a 64 bit long string.
- Take the 7th most significant bit and flip it: if it is a 0 we set it to 1 and vice versa.

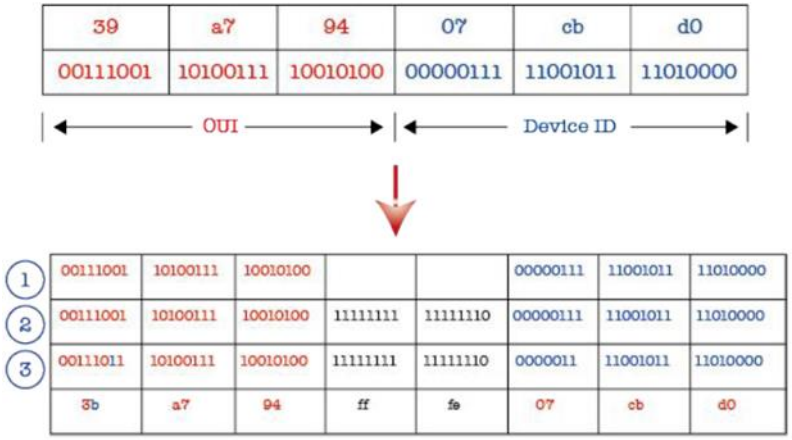


Figure 26: EUI-64 Algorithm

The MAC address was 39:a7:94:07:cb:d0 and the interface identifier obtained is 3ba7:94ff:fe07:cbd0.
The complete link local unicast address will be FE80::3BA7:94FF:FE07:CBD0.

Importance of the role of Link Local Unicast Address in IPv6

It is used in many configurations' routines of IPv6.
It is used as a source IPv6 address before a device gets one dynamically.

SLAAC (Stateless Address Auto Configuration)

The Link Local Unicast IPv6 address is used as source IPv6 address before a device gets one dynamically.
The host connects for the first time to the network, it generates its Link Local Unicast IPv6 Address and makes the network to be aware that it is there.
It sends a *ICMPv6 Router Solicitation* with source its Link Local Unicast Address (or the unspecified address, look later) and as destination address is the all-router multicast (FF02::2), so these messages are going to be delivered to all the routers connected in the sub network. Then usually there is one special router that replies

with a *ICMPv6 Router Advertisement* that contains some information about the network with as source address the router link-local ipv6 address and as destination address the all-hosts multicast, and the host will receive thanks to this message information used to generate a valid GUA for that network.
In this exchange of messages, the host already use an IPv6 Address that is its link local unicast and will use the link-local of the router as default gateway.

Routing Protocol Messaging
Routers exchange routing messages using the link local addresses.

```
R1# show ipv6 route ospf

O   2001:DB8:CAFE:2::/64 [110/657]
     via FE80::2, Serial0/0/0
O   2001:DB8:CAFE:3::/64 [110/1304]
     via FE80::2, Serial0/0/0
O   2001:DB8:CAFE:A002::/64 [110/1294]
     via FE80::2, Serial0/0/0
R1#
```

Figure 27: IPv6 Routing table

Entries in the Routing Table
Routers use the Link Local address as the next-hop address in the routing table, because they must be part of the same network.

Considerations and conclusions on IPv6 Link Local
Most of the automatic mechanisms that are in IPv6 relies on Link Local addresses and this is the most interesting features of IPv6.
Typically, the Link Local Addresses are almost always created automatically using the previous quoted algorithm.
They usually starts with FE80 (not mandatory), the following 48 bits are usually 0s, and then the Interface ID, for example: fe80::284:edff:febe:f528.
The Interface ID is typically generated using the previous algorithm starting from the MAC address, or randomly, or by setting it manually.
The only requirement is that the IPv6 Link Local Unicast address must be unique in the network. Linux by default uses the UE64 Algorithm to generate these kinds of addresses.

If the host has n different interfaces, every interface needs to have an IPv6 Link Local Address. Every IPv6 Link Address of every interface of the host belongs to the same network.

Unspecified Address

The address **0:0:0:0:0:0:0:0 (::/128)** is called the unspecified address.
It will not be assigned to any node.
It indicates the absence of an address.
One example of its use is in the Source Address field of any IPv6 packets sent by an initializing host before it has learned its own address.
The unspecified address cannot be used as the destination address of IPv6 packets or in IPv6 routing headers. An IPv6 packet with a source address of unspecified cannot be forwarded by an IPv6 router.

IV. IPv6 Configurations

- Introduction to SLAAC (StateLess Address AutoConfiguration)

This is one of the most exciting features of IPv6 that in some way removes the needing of having a DHCP server in the network.
The idea is simple: when a host joins a network and sends a *Router Solicitation*, then the router will reply with a *Router Advertisement* attaching a valid GUA Global Unicast Address prefix that can be used by the host.
Stateless means that nobody knows the state of address or knows the global table of the address used in the network.
Only network administrator can care about this because wants a bit of stability; hosts do not have any worries.
In order to realize this SLAAC, it is used **the ICMPv6 Neighbor Discover Protocol NDP**.
NDP is a special set of packets of ICMPv6 that realizes this mechanism, with messages Router Solicitation, Router Advertisement, Neighbor Solicitation, and Neighbor Advertisement.
We will focus now to the Neighbor Solicitation Message and Neighbor Advertisement Message.
Those are supposed to be the counter part of ARP of IPv4.

- ARP (Address Resolution Protocol) in IPv4

There is a host A that wants to know the MAC Address of another host B, of which knows the IP Address.
Host A sends in broadcast an **ARP request** that contains its MAC Address and the IP Address of the host B.
Every host in the sub network will receive this request: in all of them, the ARP protocol will verify, by comparing the IP Address of the host and the IP Address attached in the request, if the MAC Address requested is of the current host.
The destination host B will recognize its IP Address in the ARP request and will send an **ARP reply** containing its MAC Address directly to the sender host A (so in unicast).
Notice that every host has a table where are memorized the previous responses received by the ARP protocol called **ARP Cache** in order to avoid continuously send ARP requests to the other hosts. In IPv4 its ARP over Ethernet packet.

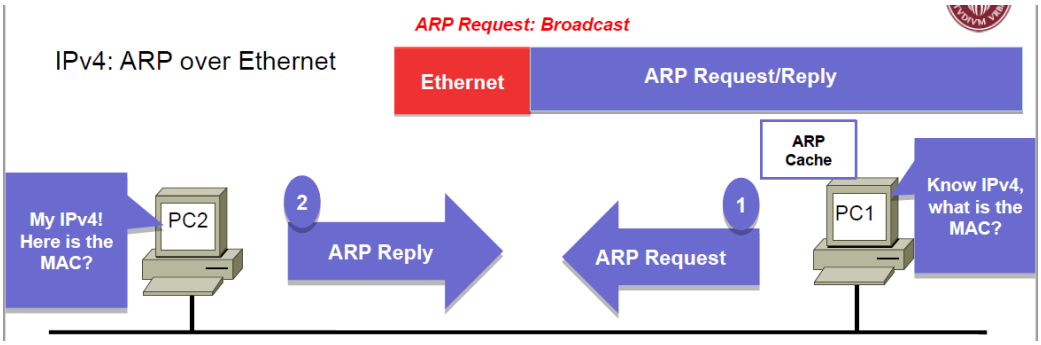


Figure 28: ARP Protocol in IPv4

- Neighbor Discovery Protocol (NDP) in IPv6

In IPv6 does not exist a broadcast address.
There is a host PC1 that knows the IPv6 address of the host PC2 and wants to know its MAC Address.
The PC1 host will investigate its **Neighbor Cache** that is the analogue of *ARP Cache*, in which it maintains the association between IPv6 addresses and MAC addresses. It sees there is not entry that can satisfy its request.
So PC1 host sends a **Neighbor Solicitation** with Source IPv6 address its Link Local Unicast IPv6 Address and destination a Multicast Address.
The host PC2 receives the Neighbor Solicitation and replies with a **Neighbor Advertisement** that contains its MAC Address with source IPv6 Address its Link Local Unicast Address and as destination the IPv6 address of PC1.

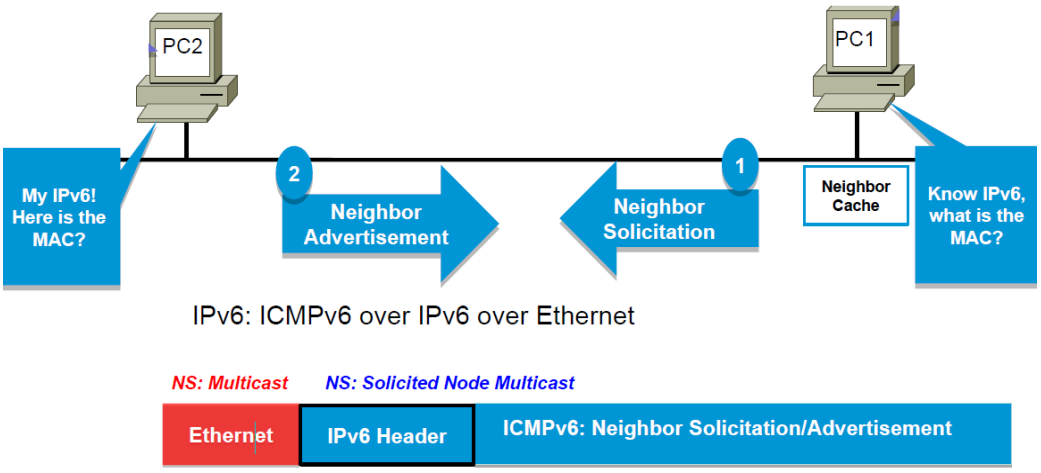


Figure 29: Neighbor Discovery Protocol in IPv6

So, in IPv6 the mechanism is an ICMPv6 that could be a Neighbor Solicitation or an Neighbor Advertisement, in an IPv6 Header that is an Ethernet packet.
So it is an ICMPv6 in an IPv6 header over an Ethernet frame.

The *IP Source Address* is a *Unicast Address* and the IP Destination Address is a *Multicast Address* that is called **NS: Solicited Node Multicast**; in Ethernet we have a **Multicast Ethernet packet**.
This helps in reducing traffic in the network.

The *Router Solicitation* and *Router Advertisement* messages are very similar and are used for dynamic address allocation; these kinds of messages are used for Router-Device messaging.

- Dynamic Address Allocation in IPv6

There is the use of the messages: **Router Solicitation** and **Router Advertisement**.
In the first step, when the host joins the network, it sends a **ICMPv6 Router Solicitation** message with destination all IPv6 routers.

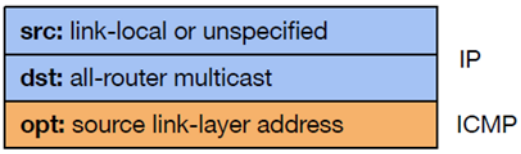
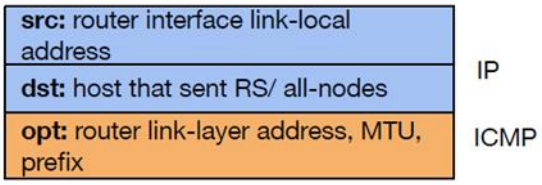


Figure 30: Router Solicitation message

The *Router Solicitation* messages are generally sent by hosts when they connect to the sub network and intended to allow host to detect what are the routers in the sub network.
The source address is link-local unicast address (FE80::/64) or it is unspecified.
The destination address is the **all-router multicast (FF02::2)**, so these messages are going to be delivered to all the routers connected in the sub network.

In the 2nd step, the router will answer with a **ICMPv6 Router Advertisement** in order to give the possibility to the device to get a GUA IPv6 Address.



The *Router Advertisement* message is generally sent periodically or as an answer to a Router Solicitation and is sent by routers.
The source address is the router interface link-local address.
The destination address could be different: if the message is periodically sent then the destination is the **all nodes multicast address** (FF02::1); otherwise it is the answer to the host that sends a Router Solicitation (FE80::/64).
The most useful option in this message is that the router that is sending this advertisement will include its link-layer address that is its MAC Address,

information about the subnetwork like MTU and also one or more network prefixes depending of the network.

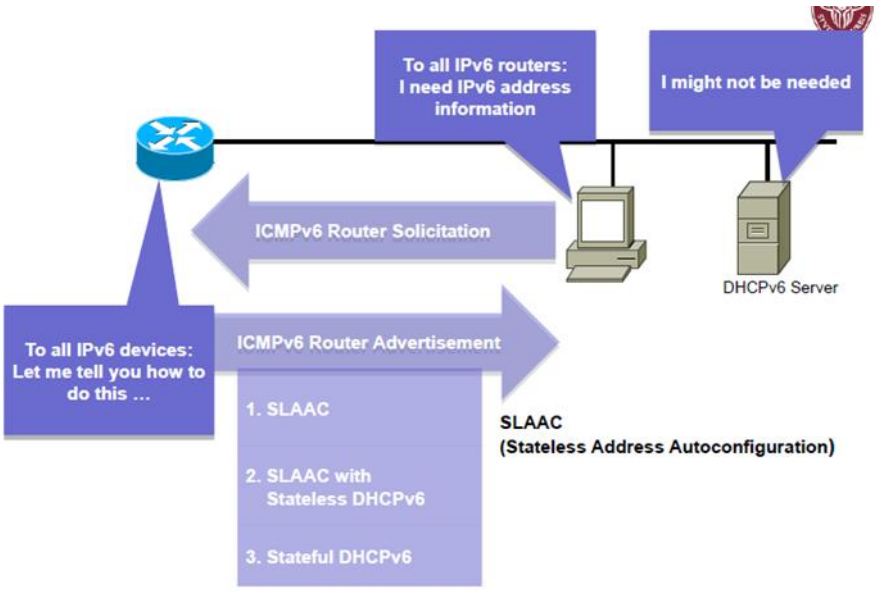


Figure 31: Dynamic Address Allocation in IPv6

The Dynamic Address configuration in the ICMPv6 Router Advertisement could be done in three ways: **SLAAC (Option 1)**, **SLAAC with Stateless DHCPv6 (Option 2), Stateful DHCPv6 (Option 3)**.

- SLAAC

SLAAC presumes no server DHCPv6, and it is the default option on Cisco Routers. There is not the maintaining the assignment of the IPv6 addresses, nobody assigns the addresses and hosts just choose their own addresses.
In this case the idea is that the ICMPv6 Router Advertisement sent by the Router is everything the host needs to set the GUA, because it contains the Prefix, Prefix Length and the Default Gateway. *But no DNS will be provided*.
The Router Advertisement has as Destination Address the **FF02::1** that is a special Multicast Address meaning **"All IPv6 devices"** and as Source Address its **FE80::1** that is its Link Local Unicast Address.
This last one is a Static Link Local, that is a very common choice for Router.
In the RA is included the Prefix and the Prefix Length.

The host, once received this information, can generates by itself its Global Unicast Address GUA using the Prefix received and by generating the Interface ID using one of the different methods (manually, EUI-64 or Random 64 bit).
The RA in this case has the two flags O = 0 and M = 0.

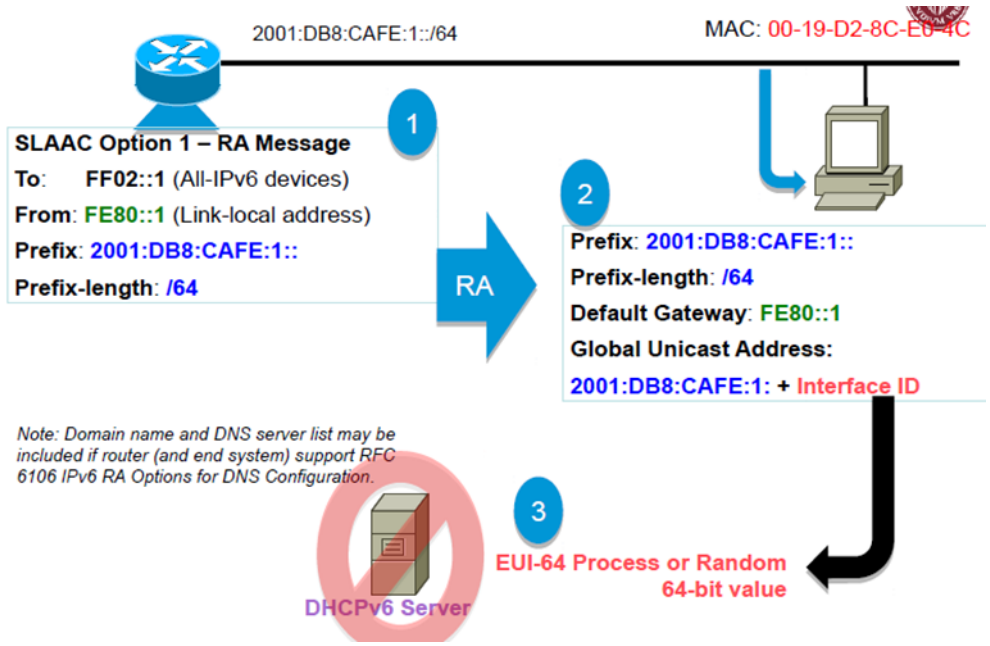


Figure 32: SLAAC in IPv6

Privacy threats with generating Interface ID

The problem with the EUI-64 Algorithm for generating the Interface ID is that the privacy of the host is compromised, because it will be the same in every network the host will join. In this way a host *could be traceable*.
Some OS by default uses EUI-64 like Windows XP and MAC OSX, while others like Linux uses a Random-64 bit.

So, an option could be to use a Randomly Generated Number and for the SLAAC there is a **Privacy Extension** that was created exactly for this, to hide who is the real host behind an IP Address.

Another alternative is the **Temporary Address** that is when a host generates a random address for some specific time (1 day or 5 minutes, how much time we want) and then just drop it and generate a new one.
The idea is to provide additional addresses that have a short lifetime and are used as source addresses when originating connections.
It is common to have multiple temporary addresses to make sure existing connections can continue while a new temporary address is created for new connections.

- DAD (Duplicate Address Detection)

SLAAC is stateless, no entity maintaining a state address-to-device mapping.
Even if the space is very huge, maybe another host has already taken the same IPv6 address another host has chosen.

The **Duplicate Address Detection DAD** is a special procedure used to guarantee that the address is unique.

The DAD works in this way: a host A has chosen its IPv6 Address X and now must be sure that it is the unique one in the network that had it.
To check this, host A sends a *Neighbor Solicitation* with Destination its chosen IPv6 Address X and if somebody answers with a *Neighbor Advertisement* then the IPv6 Address X is already used and must be used a new one.
Otherwise, if no answer is received, the X IPv6 address is not used.
In order to do this check, *the host has already done the Link Local Unicast assignment*. So the Router Solicitation and Router Advertisement messages were exchanged already.
The DAD is only the Neighbor Discovery Protocol used to detect duplicate IP addresses.

- SLAAC plus Stateless DHCPv6

The DHCPv6 does not maintain the state of the addresses. There is not the maintaining the assignment of the IPv6 addresses, nobody assigns the addresses and hosts just choose their own addresses.

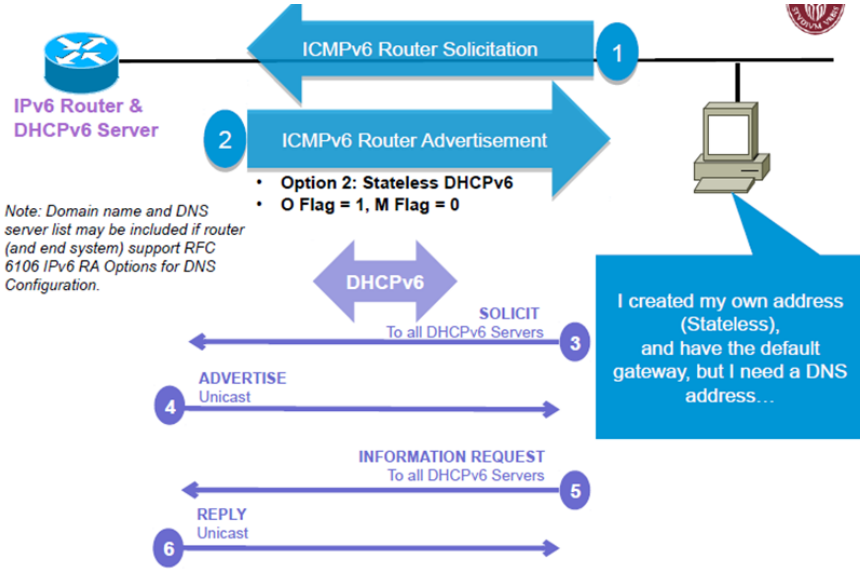


Figure 33: SLAAC plus Stateless DHCPv6

The mechanism like before: host sends a *Router Solicitation*, the router replies with a *Router Advertisement*.
In this last message RA, there are two special flags **O** (**Options**) and **M** (**Managed Address Configuration**): in this case O = 1 and M = 0.
Host however *needs to query DHCPv6 Server for asking other options*.
So, the host can create stateless its own IPv6 address and has received the the default gateway address, but has to ask to DHCPv6 Server to have other information like the **DNS address** and the domain name.
So it has to send a **DHCPv6 Solicitation** with Destination Address a special Multicast Address that is the one for **"All DHCPv6 Servers"**.

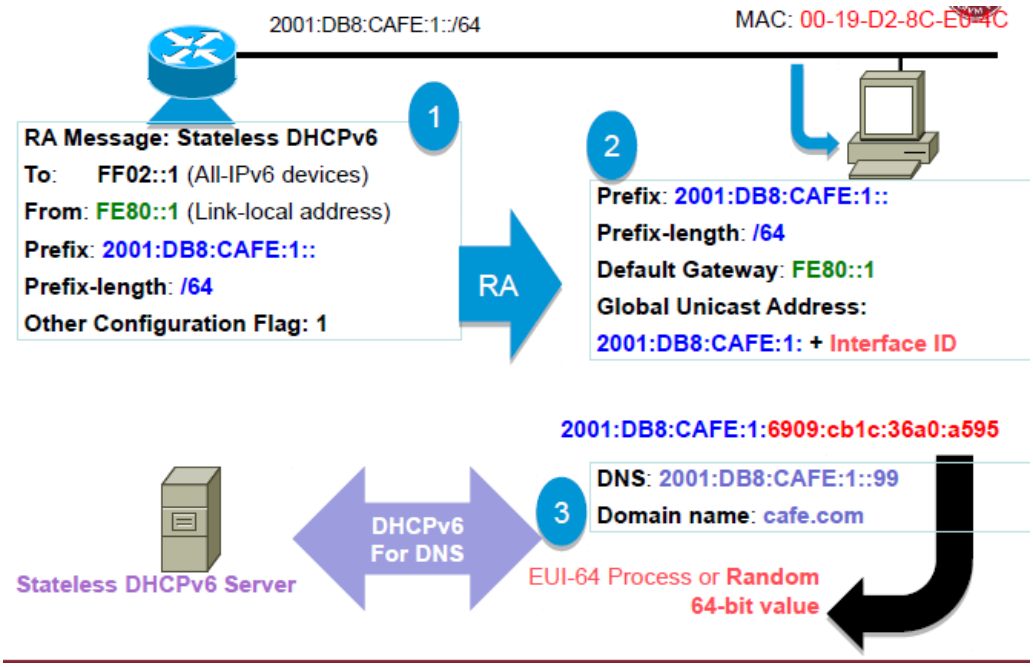
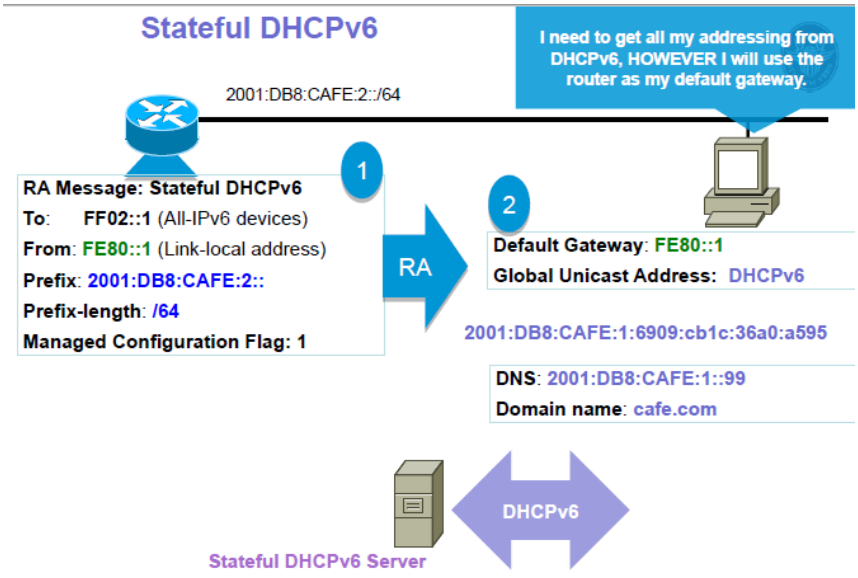


Figure 34: All SLAAC with DHCPv6 Procedure

- ## Stateful DHCPv6 Server

In this case the addresses are assigned and managed by the DHCPv6 Server. The mechanism like before: host sends a *Router Solicitation* and the router replies with a *Router Advertisement*.
In this last message RA, there are two special flags **O (Options)** and **M (Managed Address Configuration):** in this case O = 0 and M = 1, this means that the host *needs to contact a DHCPv6 Server in order to get the IPv6 Address GUA, DNS and domain name.*



The only information that the host received from this RA is the **Default Gateway Link Local Unicast Address**.

Figure 35: Stateful DHCPv6

Once it knows of the presence of the DHCPv6 Server, host immediately sends a solicit to it; this solicit has as source address it link-local address and as destination address the link-local multicast address **FF02::1:2** where the 1:2 is the group of **all DHCP servers**.

The DHCP Server communicates its offer to the Host A with an Advertise using as source its link-local address and destination the Unicast of the Host A.
At the end, the host accept the request and gets the reply from the server.
This is exactly similar to the DHCP in IPv4.

- ## DHCPv6 Prefix Delegation Process

The **DHCPv6 Prefix Delegation** is a new feature of IPv6.
It is common to ask to a DHCP Server for a single IP Address; this is the standard behavior that was already in IPv4.
Now with the DHCPv6 Prefix Delegation you can request for a whole prefix that you can use as you prefer.
The idea is that when the router is connected to the network, then it will ask to the ISP not for a single IPv6 Address but for a whole prefix so that it is possible to use it for our network and we can split it as we prefer.

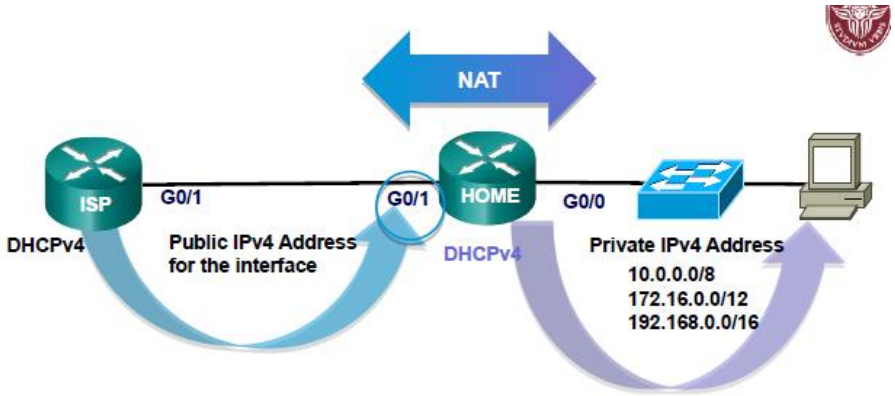


Figure 36: DHCPv4 and Private Addresses in IPv4

In DHCPv4 usually the HOME router usually asks to the **DHCPv4 ISP Server** for a *Public IPv4 Address* that will be used as a public IPv4 address; while inside the home network we will use the **Private Address Space**, so the Private IPv4 Addresses 192.168.0.0/16 or the others in the private ranges Private IPv4.
The **NAT (Network Address Translation)** is used in this context.

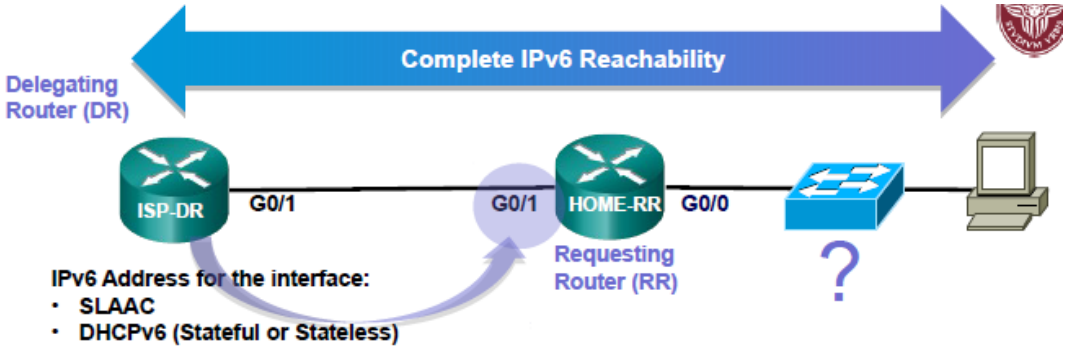


Figure 37: Complete IPv6 Reachability

In IPv6 we want to introduce the **Complete IPv6 Reachability** that means we want to give to all the hosts a Global Unicast Address.
The idea is that the **Home Router (Requesting Router RR)** will ask to the **ISP-DR (Delegating Router)** and it will give to the RR first the IPv6 Address for the public interface of the RR.
Then the RR will also send a **DHCPv6-PD Request** to the ISP-DR asking for a prefix for its internal network; the answer is a **DHCPv6-PD Reply** containing the **IPv6 prefix** that the HOME-RR can use inside the home network, and also the DNS and the domain name.
The HOME-RR will then communicate this IPv6 prefix through a Router Advertisement to all the hosts inside the LAN network.
It is a network that has been delegate by the ISP to the home router.

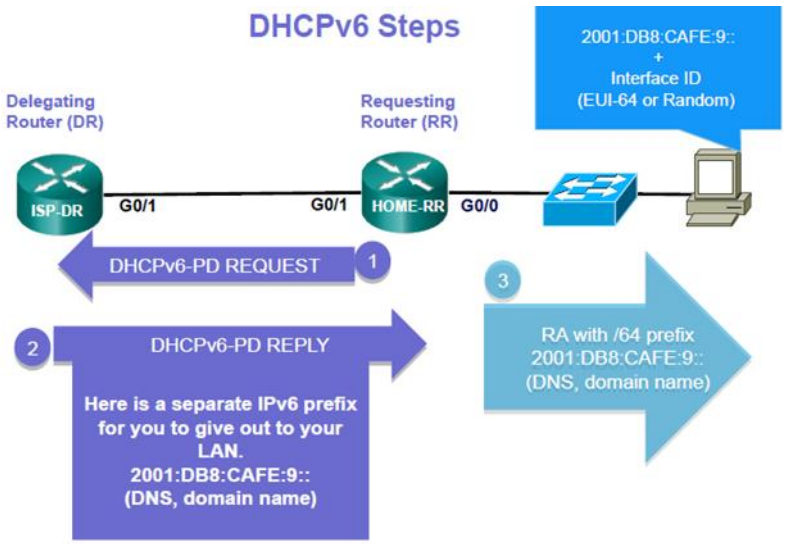


Figure 38: DHCPv6 steps in Prefix Delegation

V. IPv6 more in details

- More about IPv6 addresses

The IPv6 Addresses are divided into Unicast, Multicast and Anycast.

Anycast Address
An IPv6 anycast address is an address that is assigned to a set of interfaces that typically belong to different nodes.
Anycast addresses *are syntactically indistinguishable from unicast addresses*, because anycast addresses are allocated from the unicast address space.
A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one).

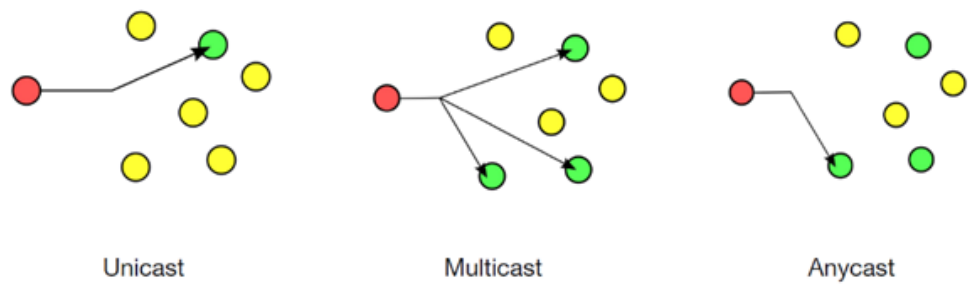


Figure 39: Conceptual difference between IPv6 types of addresses

Unicast Address
The most important groups of unicast addresses ones are the Global Unicast and Link Local, we've already talked about them.
Then we have:

- **Loopback ::1/128** : the loopback address, also called localhost, is probably familiar to you. It is an internal address that routes back to the local system. The loopback address in IPv4 is 127.0.01. In IPv6, the loopback address is 0:0:0:0:0:0:0:1 or ::1.
- **Unspecified ::/128** : the unspecified address is 0:0:0:0:0:0:0:0 .
  You can abbreviate the address with two colons (::).
  The unspecified address indicates the absence of an address, and it can never be assigned to a host. It can be used by an IPv6 host that does not yet have an address assigned to it. It cannot be used as destination.
- **Unique Local FC00::/7 :** they are used to move network in another place to keep the same network addresses.
- **Embedded IPv4 ::/80**: used with transition mechanisms to have networks that can coexists with both the IPv4 and the IPv6.

In IPv6 we do not have broadcast address, but there is the presence of multicast addresses.

- ## Multicast Addresses

Multicast address is intended to be used for one-to-many communication.
It is an identifier for a set of interfaces and a packet sent to a multicast address is delivered to all interfaces identified by that address.
An example of their use is for all services that are broadcasting (web radio, live streaming, and so on).
This is exactly the same mechanism of 224.0.0.0/4 in IPv4.
Multicast are divided in two types:

- Assigned (**FF00::/8**)
- Solicited-Node.

A multicast packet has always a source that is a unicast address.

Assigned Multicast Addresses

The Multicast address must be the destination of an IPv6 packet and not its source that must be a Unicast.
There is the concept of the multicast group which an host could be part of.
IPv6 multicast assigned addresses have the prefix **FF00::/8**, so it can go from FF00 to FFFF.

Figure 40: Multicast Range

The space reserved for the multicast is the 1/256th of the entire IPv6 address space. The only first 8 bits are always set 1 and correspond to **FFxx::/8**.
The 3rd hexadecimal digit represents the **Flag** and the 4th hexadecimal digit represents the **Scope**.

The **Scope** is 4-bit field and means how far a packet can be delivered: from very close to very large, so from only local to over the whole Internet, so to limit the scope of the multicast group. It is used to define the range of the multicast packet.

So, the scope (the 4th hexadecimal digit in the first group of 4 hexadecimals) could be: *0 Reserved, 1 Interface-Local scope, 2 Link-Local scope, 5 Site-Local scope, 8 Organization-local scope and E Global scope*.

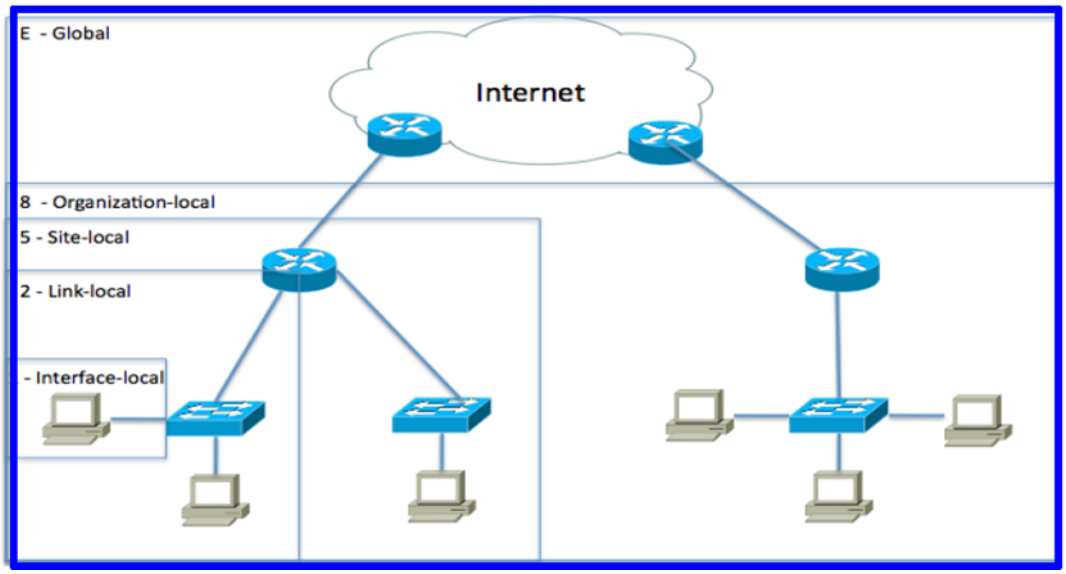


Figure 41: Scope of a Multicast Address

The 3rd hexadecimal represents the **flag** and is composed by 4 bits field.
Flags is a set of 4 flags: **0 | R | P | T**.
The first bit is always 0.
R and **P** are used for special purposes and are set to **0**.
If **T** is set to 0 then it is a **multicast permanent address (well-known)**, we will see that in each sub network there is a multicast group that represents all the routers in the sub network or all the hosts in the subnetwork or so on.
If **T** is set to **1** then it is a **multicast non-permanent address**, they are transient or dynamically assigned.
For example FF18::CAFÉ:1234, used for a multicast application with organizational scope.
So the flag, the 3rd hexadecimal could be 0 or 1:

- 0 is for the permanent, FF0x::/8 is the path of a multicast permanent address.
- 1 is for the non permanent.

The table below will show the common well-known multicast permanent addresses.

Prefix	Flag	Scope	Predefined Group ID	Compressed Format	Description (IPv6 assumed)
FF	0	2	0:0:0:0:0:0:1	FF02::1	All-devices
FF	0	2	0:0:0:0:0:0:2	FF02::2	All-routers
FF	0	2	0:0:0:0:0:0:5	FF02::5	OSPF routers
FF	0	2	0:0:0:0:0:0:6	FF02::6	OSPF DRs
FF	0	2	0:0:0:0:0:0:9	FF02::9	RIP routers
FF	0	2	0:0:0:0:0:0:A	FF02::A	EIGRP routers
FF	0	2	0:0:0:0:0:1:2	FF02::1:2	DHCP servers/relay agents

Figure 42: Multicast Well Known Addresses

The last hexadecimals groups values in this case typically represents the group to which the address will refer:

- 1 : All-Devices
- 2 : All-Routers
- 5, 6, 9 and A : Routers with a specific routing mechanism
- :1:2 DHCP servers and relay agents.

We can also change the scope and we can obtain for example **FF02::2** is the multicast address that refers to *the All-routers in the Link Local scope*, while **FF05::2** is the multicast address that refers *to the All-Routers in the Site-Local scope*.

The Multicast Addresses are for example used:

- in the ICMPv6 Router Advertisement that has a destination address **FF02::1** (All IPv6 Devices in the Link Local, including routers).
- in the ICMPv6 Router Solicitation that has a destination address **FF02::2** (All Routers in the Link Local).

There is a big difference from the Broadcast Address on IPv4 is that in IPv6 at Layer 2, in IPv6 in the frame of the Ethernet there will be a **special MAC Address** and not the Broadcast MAC Address **FF:FF:FF:FF:FF:FF**.

- ## Relay Agent in IPv6

A Relay Agent is a special device on the network that is supposed to forward DHCP requests and replies crossing the routers.
In IPv4 the DHCP server is only supposed to be used in a single subnet, in a local network.
In IPv6 since the network is very large and big because it is divided into subnets by default, it's very likely that there is one single server that manages the whole network.
So, even there are several different subnets, they can make a reference to different links, the idea is that in order to able the DHCP packets to cross the routers to use these special devices called relay agents.
With relay agents it is possible to serve several different subnets with one single DHCP server.
This is mandatory because when using DHCP, the request needs the use of Link Local Address in the packet, that cannot go out the link.
The DHCP server is outside the link and without the relay agents it is not possible.

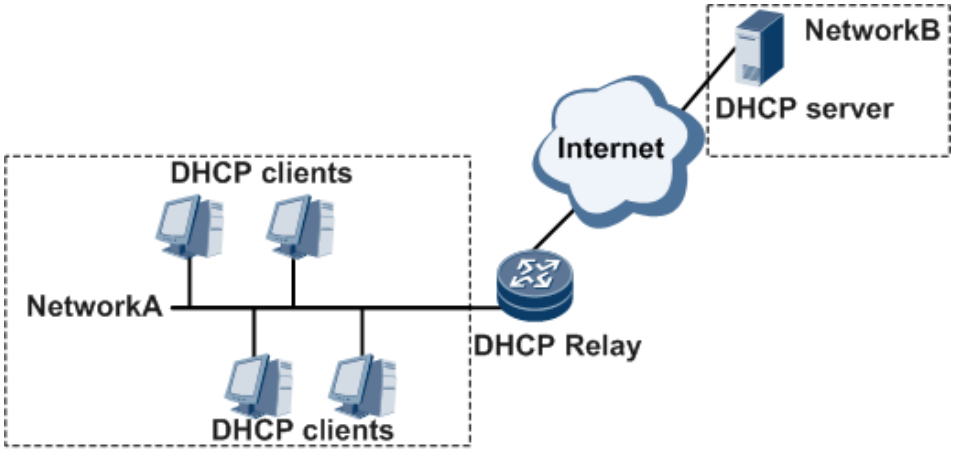


Figure 43: DHCP Relay Agent

The Relay Agents are only intermediate hosts, that make possible the DHCP requests to reach DHCP Server that is not reachable in link-local.

- ## Solicited Node Multicast Address

A **Solicited-Node multicast** address is an IPv6 multicast address used by the Neighbor Discovery Protocol to verify whether a given IPv6 address is already used by the local-link or not, through a process called DAD (Duplicate Address Detection).
This allows NDP to assign IPv6 addresses to hosts using SLAAC (IPv6 Stateless Address Autoconfiguration) without the risk of assigning addresses already in use.
The Solicited-Node multicast addresses are generated from the host's IPv6 unicast or anycast address, and each interface must have a Solicited-Node multicast address associated with it.
NDP sends out a Neighbor Solicitation message to the Solicited-Node multicast address of the IPv6 unicast or anycast address it plans to assign using SLAAC, and if a host is present in that group, it will respond with a Neighbor Advertisement message (ICMPv6 Type 136), and NDP will know that the IPv6 unicast or anycast address it is trying to assign is already in use.

A Solicited-Node address is created by taking the **least-significant 24 bits** of a unicast or anycast address and appending them to the prefix **ff02::1:ff00:0/104**
Assume a host with a unicast/anycast IPv6 address of **fe80::2aa:ff:fe28:9c5a**.
Its Solicited-Node multicast address will be **ff02::1:ff28:9c5a**.

- IPv6 Header

In the IPv6 header there are less fields then IPv4; it is simpler.
The IPv6 Header is structured in a 64-bit division, and also all the elements are based on this boundary of 64-bit; this because most of the CPUs are working with 64-bit memory.

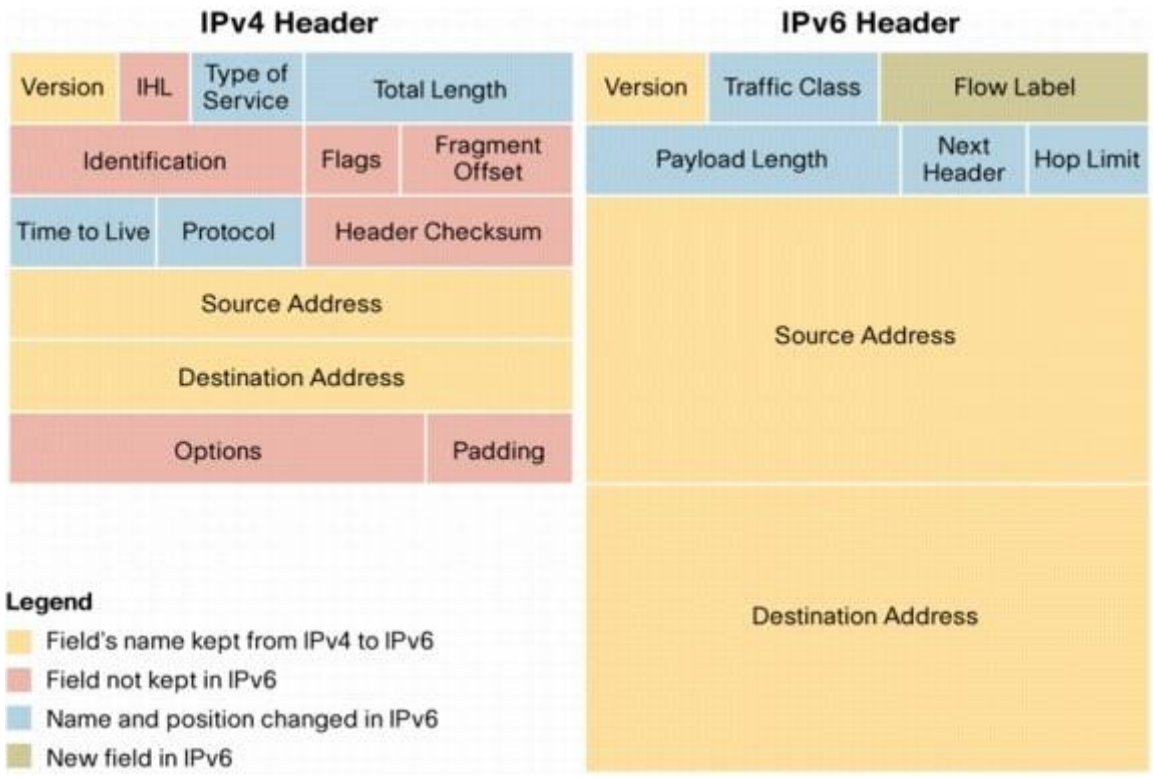


Figure 44: IPv4 and IPv6 header differences

The IPv6 headers have a **fixed size**; this was not true in IPv4 where we have the two fields related to the size of the header IHL and the size of the entire IPv4 packet that was the Total Length. IPv4 has instead a variable length header.
This was done because IPv4 packet could have a variable length to the fact the Options could have a variable size from 0 to 40 Bytes.

The **Source** and **Destination** Address are 16 bytes each one, and so they are 32 bytes long. All the other fields all together have a size of 8 bytes.
The big difference is that in IPv6 the **header size is fixed** and has a size of 320 bit or **40 bytes**.
In addition to this in order to be able to use some options, it is included a new mechanism that is based on the **"Next Header"** field, and so if there are some extensions they will be part of the payload and not of the header; this is way the header has a fixed size.

A fixed size header is an important feature because it improves a lot the efficiency of the routers. Because in this way, almost always, they are not supposed to read the packets data and process them with some operation, they have the only task (that is the meaning of their name) to route the packets.

Differences IPv4 – IPv6

- *IHL* does not exist in IPv6; it was used in IPv4 to know the length of the IPv4 header in 32-bit words including Options or Padding.
- *Type of Service* is related to type of service or quality, this field was renamed in IPv6 and is called *Traffic Class (1 Byte)*.
- *Flow Label* is a new field in IPv6 *(20 bit)*.
  It is used like a tag for the packets to know to which stream/flow they belong to. It is a label for processing the packets belonging to the same flow: for example, same flow in the same path, many systems set flow label for packets that belong to different TCP sessions. It is the same logic of transport layer to differentiate different flow of data using different ports.
  Not commonly used.
- *Payload Length (2 Bytes)* represents the bytes of the payload without the main IPv6 Header.
  The payload length includes the extensions headers length and data length.
- IPv4 uses different fields for fragmentation and reassembly: *Identification, Flags and Fragment Offset*.
  The fragmentation in IPv4 was done directly by routers every time the packet needs to be fragmented; in IPv6 intermediate devices like IPv6 routers do not perform fragmentation.
  In IPv6 routers simply notify to the sender that the packet sent does not match the *MTU (Maximum Transmission Unit)*.
  So the sender is responsible to fragment the packet if needed using a special extension header.
- IPv4 used the *Protocol field* to specify protocol used as in the payload (in the encapsulation, the protocol of the upper layer encapsulated in the payload). The analogue is the **Next Header** in IPv6.
  Common values for protocols are:
  6 TCP, 17 UDP, 58 ICMPv6, 88 EIGRP, 89 OSPF.
  We can extend this functionality by saying that if the value does not match the common values, we can expect that after the standard header there some extensions.
- *Time To Live TTL* is a field of IPv4, in IPv6 is named *Hop Limit* that actually has a meaning that is more coherent with the process. It is used to avoid a

packet to be forwarded forever if there is a misconfiguration in the network. Every router in path decrements hop limit by 1; it is set by the source.

- *IPv6 Source and Destination* are 128 bit each other.
- IPv4 uses the *Header Checksum* but some upper-layer protocols UDP and TCP already have a checksum.
  So, in IPv6, the UDP checksum is mandatory given that this field does not exist anymore in IPv6.
- *Options and Padding* are used in IPv4, not used in IPv6 (options are carried using extension headers).

• IPv6 Extension Header

The Next Header can specify two different things:

- the protocol of the upper layer carried in the data portion of the packet.
- the presence of an extension header.

Extensions headers are optional and follow the main IPv6 header.

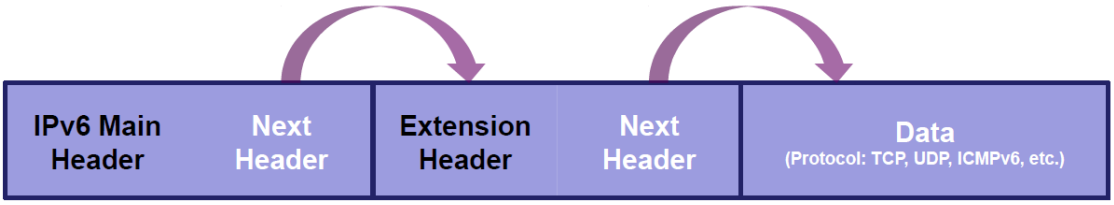


Figure 45: IPv6 Header with Next Header

So the structure could be like this:

1. the IPv6 Main Header with its field Next Header;
2. the field Next Header is pointing to an Extension Header;
3. the extension header will have its own Next Header
4. the Next Header field of the Extension Header will point to the data of the packet.

This provides flexibility and features to the main IPv6 header for future enhance ments without having to redesign the entire protocol.

Properties of the Extension Header

Flexible

Typically, there are no extensions headers in IPv6 packets.
So, when it is needed, they are inserted in the packet, and this provide a powerful and flexible mechanism. In addition, every extension header is linked using its own Next Header Field; this is because is called "**IPv6 Header Chain**" that could be a sequence of extension headers.

Flexibility brings *introduce also complexity*: there are many problems of the use of the extension's headers.
Some of them, like the one of the fragmentations, is the most dangerous one.
Also, most security devices like firewalls or intrusion detection systems must know how to correctly interpret the extensions headers.

Fixed
The number of extension header types is fixed and standardized.
Also, the order of appearance in the packet is fixed.

Processed only at endpoints
They are supposed to be processed only at the endpoints.
The task of the routers should be in theory only to forward the packet to the destination; any other task, like fragmentation, processing of the extension headers must no be a task of which routers could be responsible.
We must remove every not necessary operation that can make the routing process not efficient in terms of time/latency and operations.
*The only two exceptions: **Hop-by-Hop** and **Routing** extensions headers that must be inspected by the routers.*

Next Header Value (Decimal)	Extension Header Name	Extension Header Description
0	Hop-by-Hop Options	Used to carry optional information, which must be examined by every router along the path of the packet.
43	Routing	Allows the source of the packet to specify the path to the destination.
44	Fragment	Used to fragment IPv6 packets.
50	Encapsulating Security Payload (ESP)	Used to provide authentication, integrity, and encryption.
51	Authentication Header (AH)	Used to provide authentication and integrity.
60	Destination Options	Used to carry optional information that only needs to be examined by a packet's destination node(s).

Figure 46: Most Common Extension Headers List

0 is the ***Hop-by-Hop options extension header***, 43 is the ***Routing extension header***, 44 is the ***Fragment extension header***.
The Routing Extension does not mean exactly that a host can specify the entire path to the destination because in this way it could be possible to perform some kind of attacks some DoS on a given link of the network.
The **ESP (Encapsulating Security Payload)** and **AH (Authentication Header)** are used to realize *IPSec*.

Figure 47: Example of IPv6 packet with extension headers

Every extension header includes another **Next Header** field in its format.
In this way it is possible to know which kind of data (another extension header or the upper layer protocol header) we are supposed to find after.
Remember that while in IPv4 the routers process options in each router crossed by the packets, in IPv6 the extensions (*except Hop-by-Hop and Routing extension*) are processed only at the destination.
If encryption is present, it will start after ESP extension header, and so also the extension headers that are after in the order.

- Order of extension headers

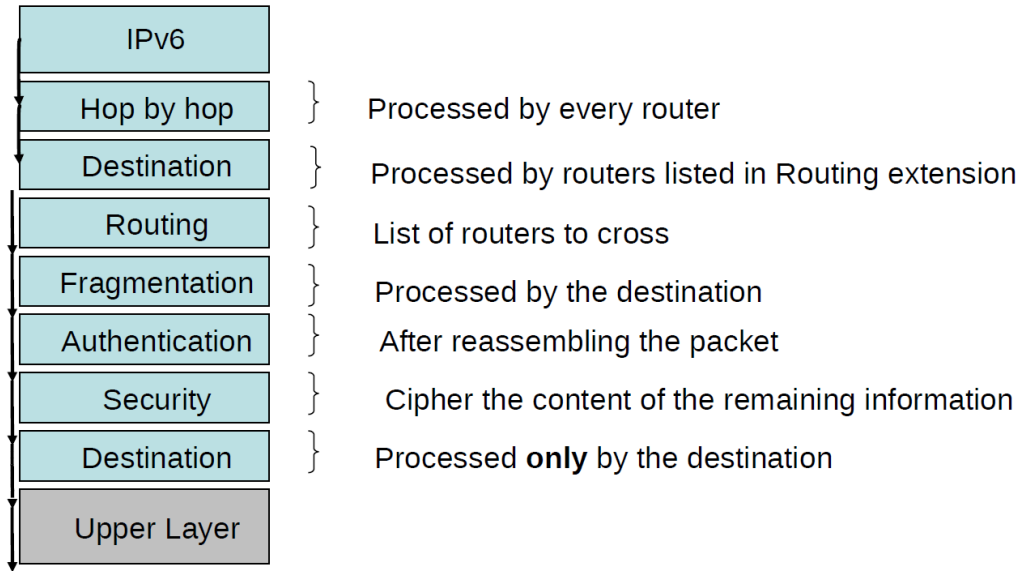


Figure 48: Extension Headers Order and use

The RFC in which the order is specified is the RFC 8600.
Destination header can be repeated twice.
The only header that can appear twice is Destination Options, because after the Security extension, because it will be encrypted, it could be read only by the host that is the final destination of the packet. Normally, it appears as the last header.
However, a Destination Options header may exist that contain options that must be examined by a list of devices specified in a source route, in addition to the destination.

In this case, the Destination Options header for these options is placed before the Routing header. A second such header containing options only for the final destination may also appear.

In case of fragmentation there is no a standard way to process, but typically a non standard rule, a convention is that only the first fragment must bring all the extension headers and the other fragments to carry payload.
Then only the first fragment must include *Extensions Header & Upper-Layer Headers* and obviously its part of Payload (First argument).
All other fragments will carry only parts of the payload

Remember that all the extensions are optional.

- Fragmentation

Each link is characterized by a parameter called **MTU** (**Maximum Transmission Unit**) that can potentially be different for every link in the path (depending on the technology used to implement that link).
MTU is a parameter that defines the largest message that can be sent over that link.
IPv4 provides a service called Fragmentation to overcome the problem when the packets have a size larger than the MTU.
If the router cannot directly forward the packet over the link, previously it has to cut the packet into different sub-parts called Fragments that have a size smaller than the MTU of the link.
In order to correctly interpret the message at the destination, the operation of Reassembly has to be done by putting the fragments together and getting the original message.
IPv4 requires that every link has a minimum MTU of 68 bytes and every internet destination must be able to receive a packet of 576 bytes in one piece or in fragments.

The difference between IPv4 and IPv6 in fragmentation are that:

- in IPv4 every single router in the path can potentially performs fragmentation;
- in IPv6 this possibility has been removed; the only node that can fragment the packet is the source node.

The advantages of these modification in the fragmentations are:

- every time a packet is fragmented in different parts, the overhead is increased because each single fragment needs to have an own header. So there is a reduction of the overhead.
- More fragmentations are done, more delay we are getting because fragmentation requires a certain amount of time.
  So there is a reduction of the delay.
- Reduction of processing.

In IPv6 the source sends the IPv6 packet with a size equal to the MTU of its direct interface.
If a router in the path to the destination notice that the MTU of outgoing link is smaller than the packet size, then the router discards the packet and sends a **ICMPv6 Packet Too Big Message** to the source indicating the MTU that it must use.
Then it's the source that must fragment the packet according to this new size.
IPv6 requires that every link have a minimum MTU of 1280 bytes, with a recommended MTU of 1500 bytes.
Path MTU Discovery uses this same process to discover the link with the smallest MTU in the path.
Because intermediate devices do not fragment packets, Path MTU Discovery is used when their links are greater than 1280.

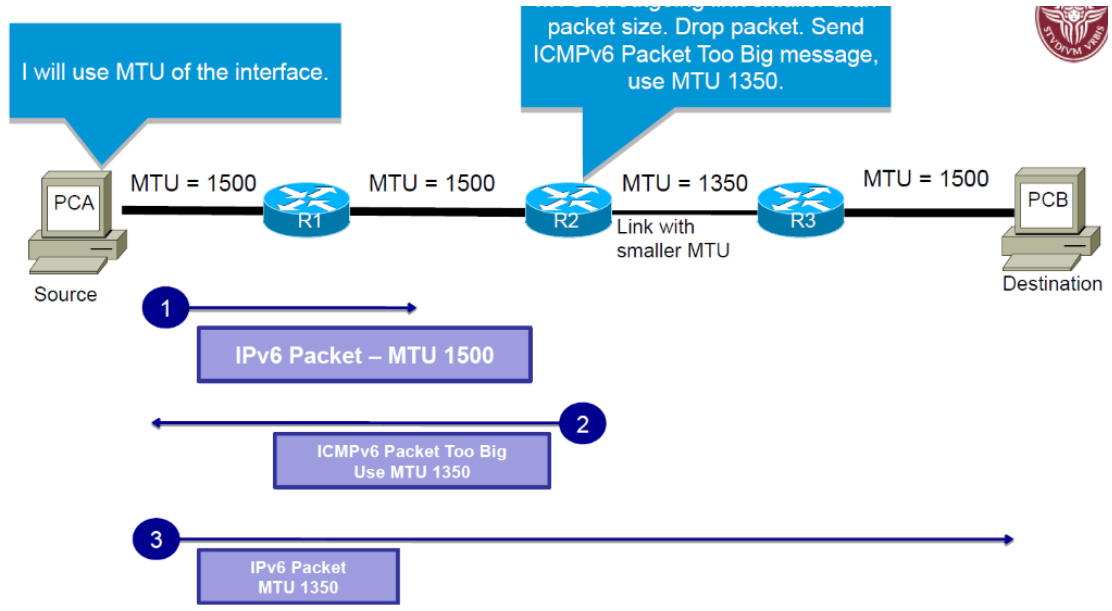


Figure 49: IPv6 do not require fragmentation by the routers

VI. Traffic regulation using firewalls

- Traffic regulation

When we want to join a network, we usually talk about routers.
The router is a device that connect two networks and it is supposed to make traffic to flow.
The idea is to introduce some kind of permission decision before allowing a packet to flow from one network to another one, that means to regulate the traffic.
We should have some kind of rules to determine if a packet is supposed to go on the other part of the network.

A **firewall** is a device that besides acting as a router, contains and implements rules to determine whether the packets are allowed to travel from one network to another.
For simplicity we talk about 2 networks, but actually a firewall can connect many different networks. Routers can also perform some kind of screening, packet filtering.

If firewalls are not used, we are exposing all the devices of the network to everyone. With a full reachability from the outside, everyone can try to access to the services running on that hosts.
So in this sense the use of firewalls **restricts the access from the outside** and prevents attackers from getting too close.
Also we can have some rules **to restrict people from leaving**, for example to exclude some portion of the network to access services on the Internet.

To achieve a big level of network security, it is possible to:

- regulate traffic that is allowed (sources, destinations, services, …); this is related to the access control;
- protect the traffic by encryption, related to confidentiality;
- monitor the traffic and monitor the hosts for “bad behaviour”.

There will be the use of other kind of devices for these different purposes:

- the regulation of traffic is done with **firewalls**;
- encryption traffic is managed with **VPNs**;
- the monitoring of hosts and traffic is done with **Host IDS (Intrusion Detection System)** and **Network IDS (Intrusion Detection System)**.

- Security strategies in Firewall design

Least Privilege

Basically, the principle of least privilege means that any object (user, administrator, program, system, whatever) should have only the privileges the object needs to perform its assigned tasks – and no more. Least privilege is an important principle for limiting your exposure to attacks and for limiting the damage caused by attacks. Every user probably doesn't need to modify (or even read) every file on your system. Every user probably doesn't need to know the machine's administrative password.

Defense in depth

Another principle of security (again, any kind of security) is defense in depth. Don't depend on just one security mechanism, however strong it may seem to be; instead, install multiple mechanisms that back each other up.
You don't want the failure of any single security mechanism to totally compromise your security. You can see applications of this principle in other aspects of your life. For example, your front door probably has both a doorknob lock and a dead bolt; your car probably has both a door lock and an ignition lock; and so on.
This means to add redundancy to the defensive measures, remove the single point of failure, find the right balance between complexity and multiplicity of defense measures. So that in order to compromise the system, the attacker has to find not only one vulnerability, but n vulnerabilities in different components.

Choke point

A choke point forces attackers to use a narrow channel, which you can monitor and control.
In network security, the firewall between your site and the Internet (assuming that it's the only connection between your site and the Internet) is such a choke point; anyone who's going to attack your site from the Internet is going to have to come through that channel, which should be defended against such attacks.

Weakest links

A fundamental tenet of security is that a chain is only as strong as its weakest link and a wall is only as strong as its weakest point. Smart attackers are going to seek out that weak point and concentrate their attentions there. You need to be aware of the weak points of your defense so that you can take steps to eliminate them, and so that you can carefully monitor those you can't eliminate.

Fail-safe stance

Another fundamental principle of security is that, to the extent possible, systems should fail safe; that is, if they're going to fail, they should fail in such a way that they deny access to an attacker, rather than letting the attacker in. The failure may also result in denying access to legitimate users as well, until repairs are made, but this is usually an acceptable trade-off.
For example, if a packet filtering router goes down, it doesn't let any packets in.

Universal participation

Everybody should be under the umbrella of this security control.

Diversity of defense

It should be used different types of defensive mechanisms.

Simplicity

Complexity introduces easier errors in configuration and so on.

- Different setups of firewall

Host-based packet filters

This is related to the protection of a single host of the network.
There is a single host that is supposed to receive the packets, and forward these packets applying a set of rules to making decisions if the packet could be received or sent.
These are for example: iptables, Windows Firewall and all the personal firewalls.
In Linux the default policy is to accept packets coming from the outside or going to the Internet.

Screening router (ACL-based)

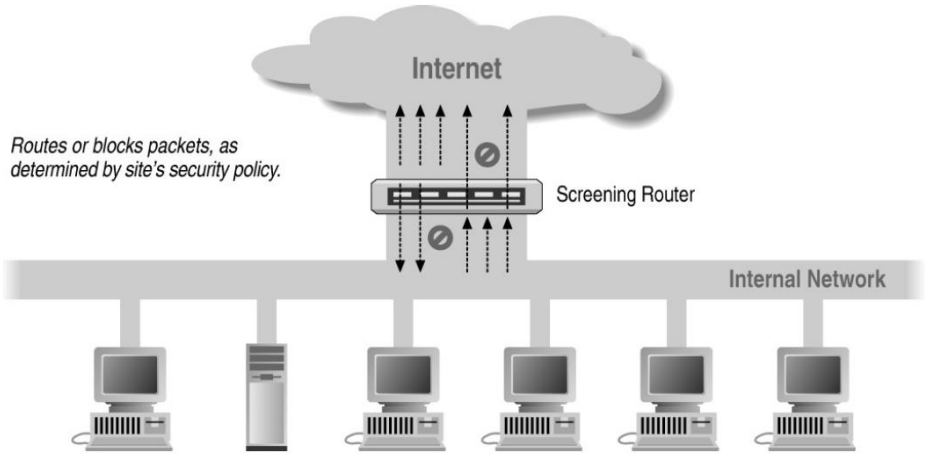


Figure 50: Screening Router

This is related for network traffic and it is not intended to be received by the single host itself. In this case there is some ingoing and outgoing traffic and we have to decide if it could go inside or outside the network.
The first level is called a **Screening Router** that is making decision using an **ACL Access Control List** based screening.
It is for example possible to list networks from which traffics could come into the internal network. The Access Control List is a list of the rights for accessing/using networks and it's a lot used in switches, routers and firewalls.
Usually distinguish between incoming and outgoing traffic, per interface and port; for example, a lists of IP addresses that can send packets to a port.
The ACL is also **stateless**: means that every packet is treated independently, without any knowledge of what was come before.
Every decision is taken on the single packet when it is received, it is not based on the previous packets received.

Dual-homed host

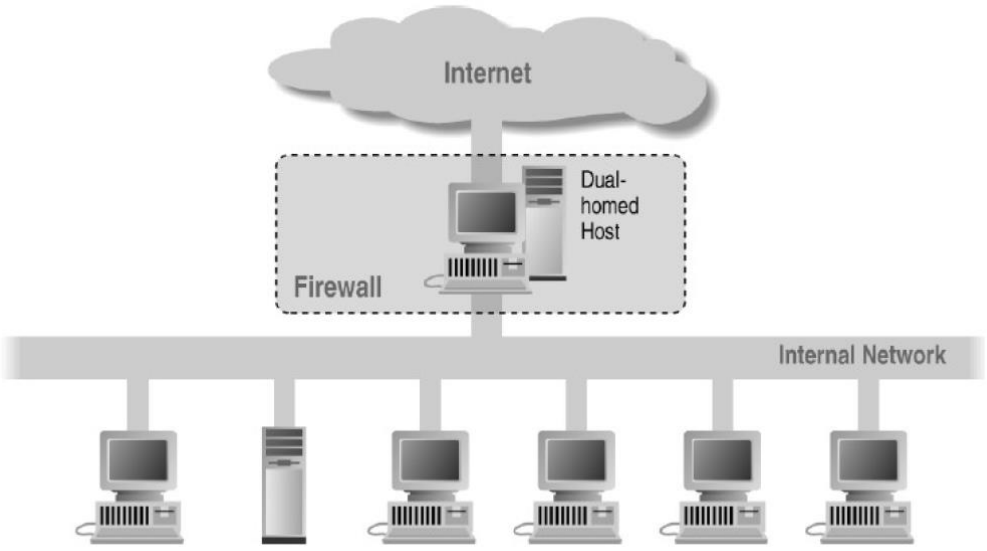


Figure 51: Dual-homed host

In this scenario the firewall is realized by a dual-homed host.
It is an host that is supposed to have 2 different interfaces, and the decision is taken to forward every single packet from the one network to the other one.
It is very similar to the concept of the router, but know it is a real host with a real operating system, like OPNsense.

Bastion host

It is like the concept of a castle, that is a really strong building suppose to face the attacks of the enemies, and this is way it is called bastion host.
It is hardened computer used to deal with all traffic coming to a protected network from outside.
Hardening is the task of reducing and removing vulnerabilities in a computer system: that means the minimum amount of only needed services to make the host run as a firewall, so by removing all other services, no additional users, stricter configurations, all not needed access controls to files, all not needed permissions, and so on by applying the least privilege principle.
The principal difference with the screening router scenario is that this one is based on ACL list, while with a bastion host it is possible to write more complex rules and to take decisions not only based on IP-stateless but also **stateful** decisions.
Stateful means that there is a log of the previous received packets/traffic, and the decision of the current packet is done also by recalling the already received traffic (for example if a packet belongs to an already accepted stream of traffic).
The bastion host is a typical candidate for realizing a VPN gateway and suitable for use as Application Proxy Gateways.

DMZ (Demilitarized zone)

DMZ is a computer host or a small network inserted as a “neutral zone” between the company’s private network LAN and the outside public network.
It is like an air-gap between the private LAN and the Internet.
This is used to realize a secure segregation of networks that host services for users, visitors and so on.

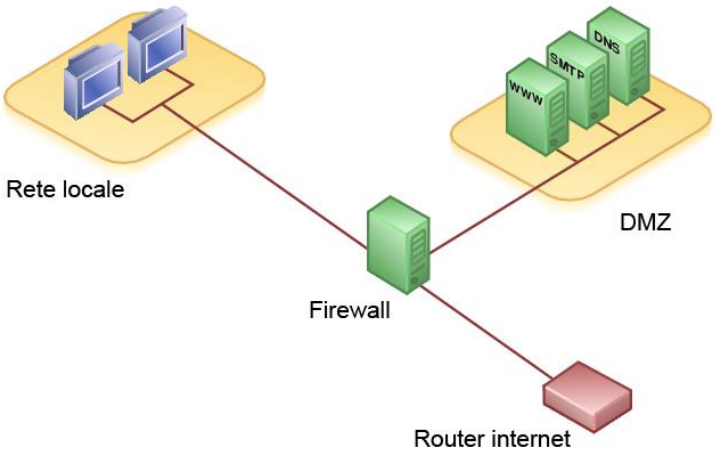


Figure 52: Example of DMZ

For example, if there are some services that are accessible from the outside of the company must be placed into DMZ and not in the private LAN, in this way there is

a DMZ separated network from the internal LAN network.
It is like having a public area in which there are public services that are isolated from the private LAN.
In this way we reduce and regulate the access to the internal private LAN.
DMZ is based on the principle of the defense in depth because there is a multilayered level of security.

DMZ as a screened Host

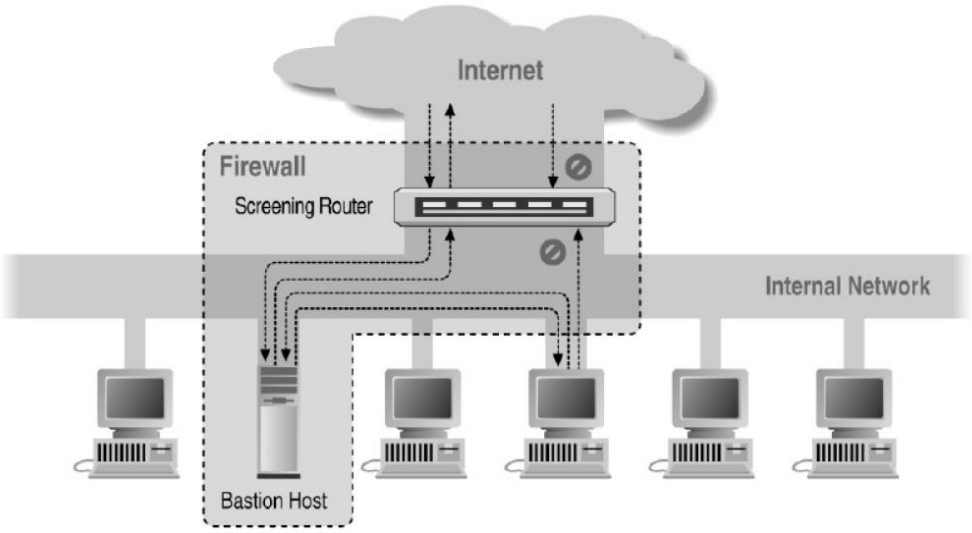


Figure 53: DMZ as screened host setup

There is a screening router that only allows the traffic from the Internet to reach only the Bastion Host. Then it is the Bastion Host to decided if the packets must be forwarded on the internal network or not.
Also any traffic supposed to go out directly from the screening router should be blocked.

Screened Subnet using two firewalls

This is the standard DMZ.
There is an intermediate network, the **Perimeter Network**, where there are two different routers, one supposed to be exposed to the Internet, the Exterior Router and one supposed to be used from the internal network, the Interior Router.
Inside there is the Bastion Host that makes the decisions to allow traffic going into the private network or the other way around.

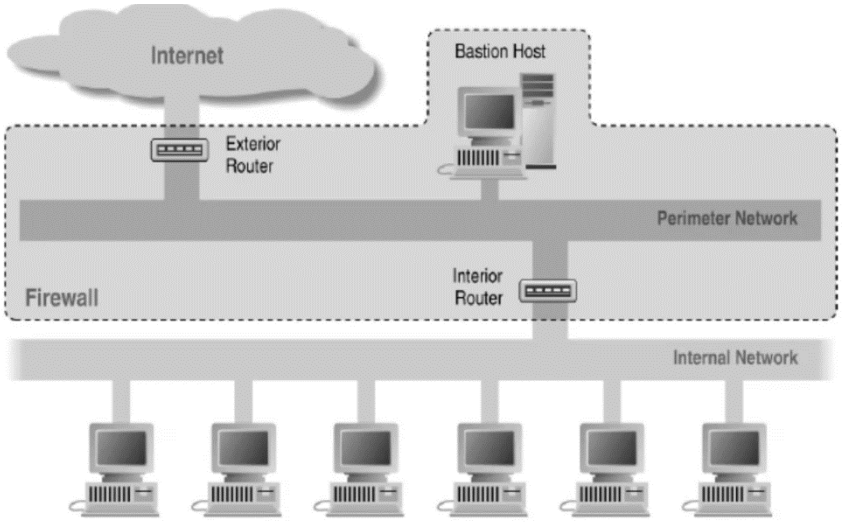
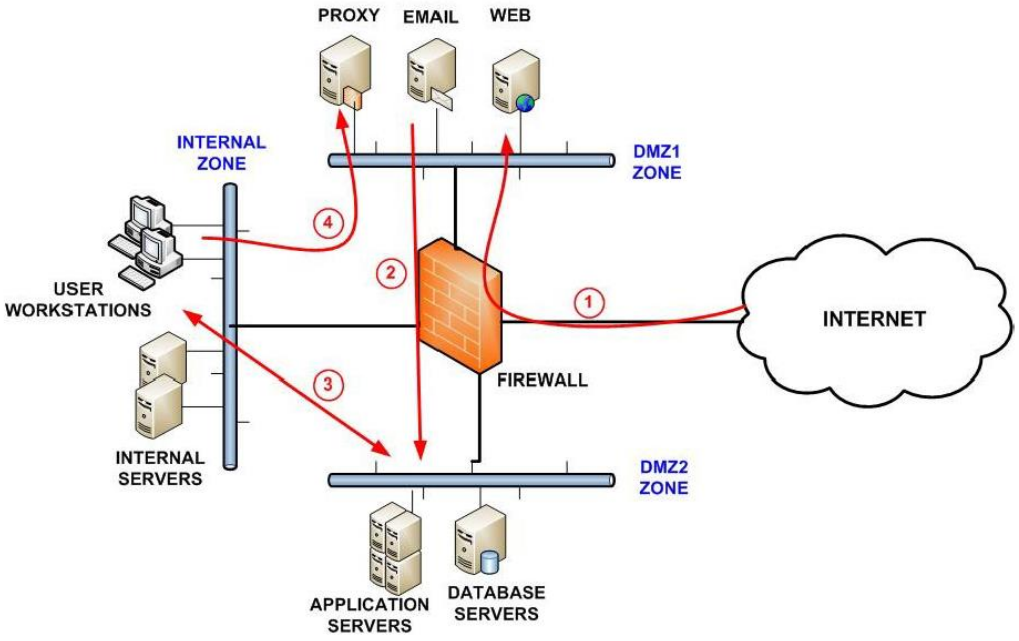


Figure 54: Screened subnet using two routers/firewalls

DMZ to segment network



There's a single firewall that realizes the segmentation of the whole network using several multiple interfaces.
In the firewall there are several rules that in some way determines the type of movements that the traffic can perform inside the three networks.
This kind of scenario introduces a lot of complexity in the kind of policies (several networks must be taken into account) and a single point of failure.

It is also possible to split different layers of DMZs, having Internet, than an External DMZ Network that can access through Main Firewall to an Internal DMZ network, that have access through Internal Firewall to the private LAN network.

- ## Packet filtering

When you want to perform network security, you want to filter ingoing and outgoing traffic.
Ingoing traffic means that interface is generating traffic, and it is incoming on that interface.
Outgoing traffic means that there is a stream of packets that the router/firewall is inserting in the interface.
It is also important to decide, when there are different places in which we can insert a rule, to place the rule *closer to the source* or *farthest from the source* this can impact also on the traffic movement.

There are some assumptions to keep:

- Security policy stating what is allowed and what is not allowed;
- It is possible to identify good and bad traffic by combining IP addresses, TCP port numbers, flags of some protocols, and so on;
- The firewall itself is immune to penetration, we need trusted system.

A **packet filter** is a **stateless** firewall, every decision is taken independently from all the other previous, every packet is individually evaluated.
Drop packets is based on source or destination addresses, port numbers or flags.
There is no context and no concept of the state of the connection.
It is possible to operate on incoming interface, outgoing interface or on both.
It is possible to check packets with fake IP addresses from outside *"ingress filtering*" and from inside *"egress filtering"*.
In this type of firewall we have weaknesses related to the absence of the concept of states and problems related to the IP fragmentation.

Packet filtering operates on Data-Link layer, Network Layer and a bit on Transport Layer in which we take into account only source port, destination port and flags.
The TCP flow is made by sequence numbers and states, and we cannot set rules for these in packet filtering, so we cannot realize some kind of protections.

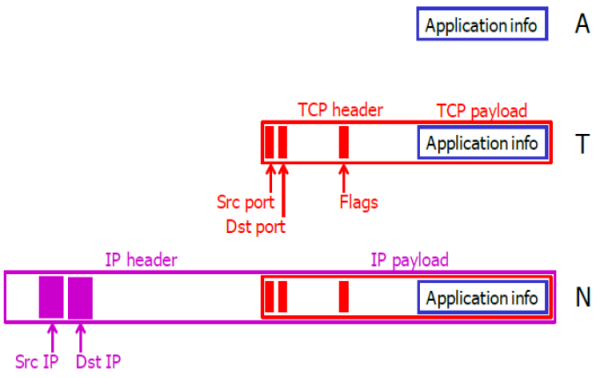


Figure 55: Packet filtering operating layers

The packet filtering is three-step process:

- Know your policy: *"Accept only HTTP traffic"*
- Translate the policy in a formal language: *"Allow packets from any source and from any port to destination host my web server and destination port TCP 80"*
- Rewrite the policy in terms of the firewall syntax:
  "iptables -A FORWARD -s 0/0 -d 192.168.1.58 -p TCP --dport 80 -j ACCEPT"

The general mechanism is that:

- Rules are check from top to bottom
- The first matching rule is applied
- One implicit rule is assumed if no rule matches: the default policy could be block everything (**closed firewall**) or allow everything (**open firewall**, like in Linux).

It is important in rules to specify the direction of the traffic, because if we allow generic traffic from port 25 for example, we cannot control the traffic coming from the outside originated from this port.
In TCP communication we can check the direction by using the TCP flag **ACK**: if there is the flag ACK in a TCP stream, then it is possible to assume that this is an answer to an already initialized communication.
In this way we can allow a TCP connection to be initialized *only* by the internal network using as source the address of the internal LAN and port the TCP port, and at the same time it is possible to allow replies to this existent stream by allowing TCP traffic on port 25 with flags ACK.

- Example of packet filtering rules

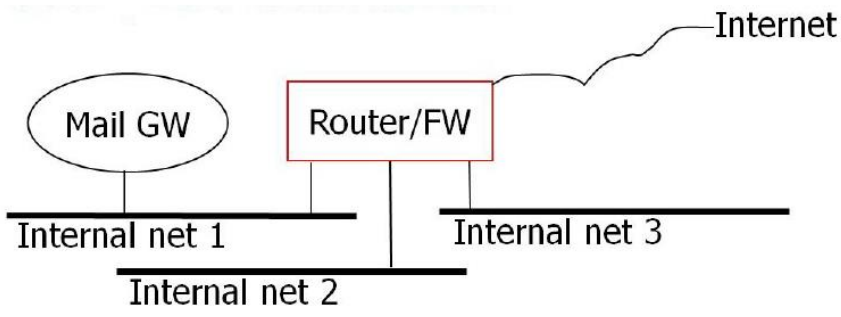


Figure 56: An example of possible network topology

There is a router/firewall with 4 interfaces, one is connected to the Internet.
Internal Net 1 is a DMZ and only host Mail Gateway, so we want this to be accessed from the outside.
We want to limit connections between Mail GW and Internet, so the Mail GW send traffic only to other partner servers.
Also limited connections allowed between Mail GW and Internal net 2 and Internal net 3.
Anything can pass between Internal Net 2 and Internal Net 3.
Internal Net 2 and Internal Net 3 can initialize connections towards Internet.

So, the rules on the **Internet interface** on the firewall are:

Action	IPsrc	srcport	IPdst	dstport	flags
block	"Net 1"	*	*	*	
block	"Net 2"	*	*	*	
block	"Net 3"	*	*	*	
allow	*	*	GW	25	
allow	*	*	"Net 2"	*	ACK
allow	*	*	"Net 3"	*	ACK

Figure 57: Rules on the Internet incoming interface

The first three rules are blocking tries of spoofing IP addresses, because they are blocking packets that are coming from the Internet with source address one of the three internal networks.
The 4th rule is accepting generic traffic coming from anywhere from the Internet to the host Main GW on port 25.
And the 5th and 6th rules are allowing the traffic coming from anywhere from the Internet directed to the network Internal Net 2 or Internal Net 3 that is an answer to an already established connection (TCP ACK flag).
The rules on the Internal Net 1 interface are:

Action	IPsrc	srcport	IPdst	dstport	flags
allow	GW	*	"partners"	25	
allow	GW	*	"Net 2"	*	ACK
allow	GW	*	"Net 3"	*	ACK
block	GW	*	"Net 2"	*	
block	GW	*	"Net 3"	*	
allow	GW	*	*	*	

Figure 58: Rules on the Internal Net 1 interface

The 1st rule allows traffic coming from the Mail Gateway host to the partners network on port 25.
The 2nd and 3rd rule allow answer from the Mail Gateway of an already established communication directed to Net 2 and Net 3 (Mail Gateway cannot initialize the connection).
The 4th and 5th rule block all other connections directed from the Mail Gateway to the internal networks.

For the Internal networks 2 and 3 interfaces rules are simple.

- Problems with packet filtering

1) Only a small number of parameters: it is easy to specify rules which are too specific or too general.
2) Payload of TCP packet is not inspected: there is no protection against attacks based on upper-layer vulnerabilities.
3) Limited logging ability (restricted on the few parameters used on the filter)
4) No authentication facilities
5) Susceptible to attacks based on vulnerabilities in implementations of TCP or IP.

Problem 1

- Example: Filtering in- and outgoing SMTP traffic. Try rules:

Rule	In/out	IPsrc	IPdst	Proto	dstport	Action
A	Inward	External	Internal	TCP	25	Allow
B	Outward	Internal	External	TCP	>1023	Allow
C	Outward	Internal	External	TCP	25	Allow
D	Inward	External	Internal	TCP	>1023	Allow
E	*	*	*	*	*	Block

- Scenario:

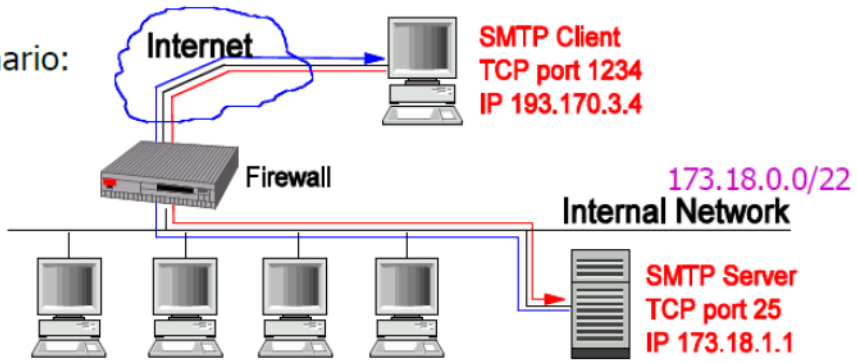


Figure 59: Filtering in and outgoing SMTP traffic, scenario

There is an internal network with address 173.18.0.0/22 with an SMTP Server with IP address 173.18.1.1 accepting on TCP port 25, and there is outside an SMTP client with IP address 193.170.3.4.

Look at the slides from Filter Rules, 1 to Filter Rules 9 for the example.

Problem with IP Fragmentation
The IP packet is fragmented in n different packets, and the length indicates the length of each fragment, the ID indicates the datagram and is common to all fragments, the offset is used as the position of the fragment in order to reconstruct the original packet, the fragflag indicates if there are other incoming fragments.
In a normal fragmentation, they are normally reconstructed in a one packet.

In an abnormal fragmentation, the fragments are not reassembled one after the other, but one on top of the other, like overwriting one on each other.
Then if you have something like a TCP header, if there is an overlapping fragment, it is possible to overwrite the TCP header and it could be a problem.
Normally, each one starts after the last one ended. However, an attacker can construct packets where fragments actually overlap, and contain the same data addresses. This does not happen in normal operation; it can happen only when bugs or attackers are involved, and attackers are by far the most likely cause.
Also, it is possible that internal host reassembles a packet with the SYN bit set because two fragment offsets are chosen in order to set the SYN bit.

- ## Stateful packet inspection

Stateful Inspection Firewalls or Dynamic Packet Filters can keep track of the established connections. Instead of having every packet that is evaluated by itself, there's the introduction of the notion of **established connection**.
When a packet is received, not only all the parameters of Source IP, Destination IP, port and so on are checked, but also can check if there is an ongoing connection and that packet belongs to this already existent connection.
In this way it is possible to solve one major problem of simple packet filters, since they can check that incoming traffic is a genuine response to a previous outgoing request to set up a connection.

The stateful firewall is based on the **connection tracking**.
With a stateful firewall we consider the Data Link Layer, the Network Layer and this time the full Transport Layer.

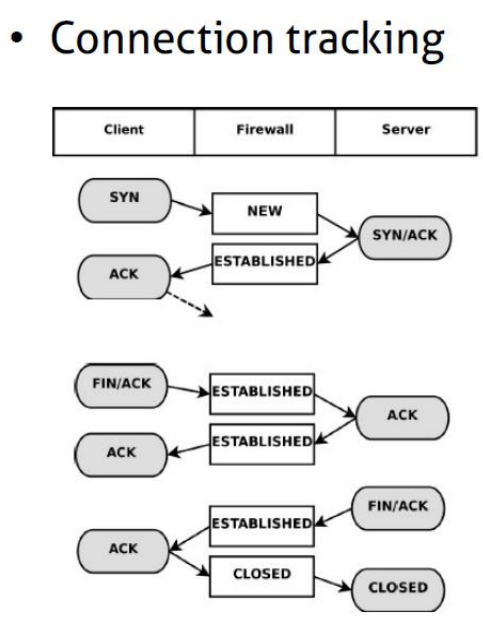


Figure 60: Connection tracking mechanism using connection states

The stateful firewall check the connection status.
When a new connection is initialized from the internal network (for example) to the outside, the state of the connection is maintained by the firewall.
With the SYN in the TCP we create a NEW CONNECTION, when there is a SYN/ACK the connection status is changed to ESTABLISHED.
If for example we want to allow packets that are answers to an already existent connection, we can take in this case the decision to allow packets for the ESTABLISHED connection.

The firewalls recalls that there is an established connection and takes the decision with respect to that state. In this case, the decision is taken not only looking at the fields of the packet, but also at the ***State Table*** of the connections.
So it is related not to the packet itself, but to the context and the history of the traffic that was already seen previously.
In TCP, connection is considered **established** when the server gives the correct SYN/ACK response. Connection is considered **closed** both parties have to close the connection by sending a TCP packet with FIN flag set before connection is considered closed**.**
The idea is that TCP protocol has states mechanism, so we can use these states to take decisions.

Remember that the connection tracking could be generalized not only for TCP but for every protocol like UDP.
For example, an internal host A sends an UDP packet with source port X to an external host B with destination port Y. Then, there is a timer, and if in this slice of time there are packets coming from host B with source port Y and destination host A and destination port X, it is possible to check in the *states table* that there was an already existent connection between the hosts A and B, and it is accepted by the firewall.
If the timer expires or the fields on the packet do not match with the previous connection, the packets are not accepted anymore.

- ## Other types of firewalls

With the stateless and stateful firewalls it is possible to match fields of packets from Data Link Layer, Network Layer, up to the Transport Layer.
If we want also to inspect the payload of the packet, then it becomes an **Application-Level filtering.**
This is for example mail filters, FTP, HTTP proxy and the proxy mechanism.
There are some pros for the app level firewalls like logging capacity, user level authentication, but also cons like lag, different firewall for different applications.

Then there is the already mentioned the **host based firewall**, that is a firewall on each individual host to protect its own single machine.
This could be used to selectively enable specific services and ports that will be used to send and receive traffic, and it could be seen as a part of the defense in depth if one of the component of the IT system is already compromised.

There are the **Circuit-level gateways (generic proxy)** also known as TCP relay. SOCKS is the de facto standard. The client connects to a proxy that relays its connections in a protocol-independent manner.

Finally there are the **Next Generation NG firewalls** that try to include additional features. Not only traffic filtering but also IDS (Intrusion Detection System), VPN gateway, Deep Packet Inspection, Traffic shaping.

- ## Common firewall weaknesses

There is no content inspection that can causes problems like software weakness (SQL Injection, Buffer Overflow, Exploits) and protocol weakness.
There is no defense against DoS attacks and insider attacks.
The Firewall failure must be prevented by using cluster of firewalls.

- ## Routed and Transparent firewall modes

In **routed mode,** a firewall is a hop in the routing process; it is a router that is responsible of its own internal networks.
In **transparent mode**, the firewall is hidden to the network topology and usually it is realized at Layer 2, and is not seen as a router hop to connected devices.
So it is possible to take decisions in a completely transparent way with respect to users and hosts, and it can be implemented bridged NICs.

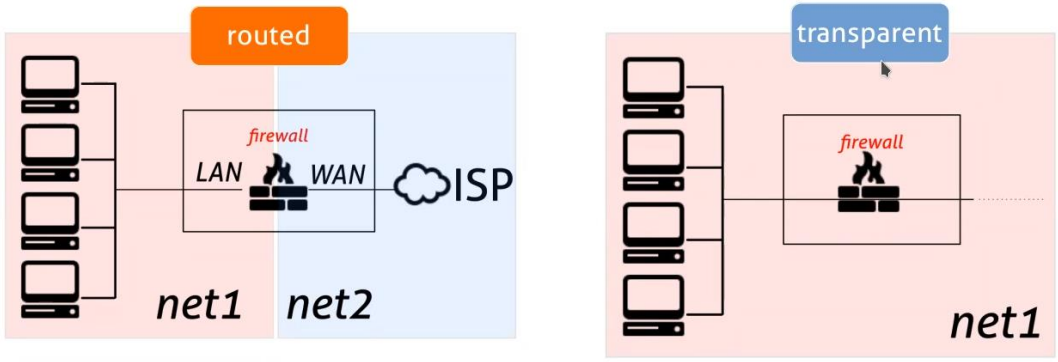


Figure 61: Routed and Trasparent firewall modes

VII. Iptables

- Introduction

It is the implementation of a packet filtering firewall for Linux that runs in kernel space. It is the evolution of ipchains and ipfw. Coming successor will be nftables.
iptables tool inserts and deletes rules from the kernel's packet filtering table.
It can also operate at the Transport layer (TCP/UDP).
Here there is a manual for iptables:
https://www.frozentux.net/iptables-tutorial/iptables-tutorial.html

- Fundamentals

All the rules are grouped in **tables**. Each table has different **chains of rules**.
Tables are made up of built-in chains and may also contain user-defined chains.
The built-in tables will depend on the kernel configuration and the installed modules. This can be specified by the parameter:

- **-t *tableName***

Each packet is subject to each rule of a table.
Packet fates depend on the first matching rule.
Once a matching rule has been applied, the other rules will not be considered but **with the exception** of a **LOG** rule and after the NAT Table it is possible to retrieve the same packet on another table.
The default tables are as follows: **Filter, Nat, Mangle, Raw and Security**.
The *Filter Table* is the default table selected if no parameter t is passed.
The filter table is used to decided if a packet could pass or not, the other tables are used to manipulate the packet.
There are the basic commands:

- Show rules list: **iptables -L** or **iptables -L -n -v --line-numbers**
- Delete all rules from all chains (flush): **iptables -F**
- Delete all rules from a specific chain***:*** **iptables -F *chainName***
- Append a rule to the chain: **iptables -A *chainName* rule**
- Insert a rule to the chain (default 1st): **iptables -I *chainName [rulenum]* rule**
- Delete a rule from a chain: **iptables -D *chainName ruleNumber***

- ## Filter Table

The Filter Table is the default table (if no -t option is passed).
There are three default chains in the default table Filter.

- **INPUT**: This chain is used to control the behavior for incoming connections to the host itself.
- **FORWARD**: packets forwarded by your server, if/when it acts as a router between different networks.
- **OUTPUT**: This chain is used for outgoing connections. Packets generated by the process on the host that is running the firewall.

The FORWARD chain only used when the machine is configured as a router (net.ipv4.ip_forward to 1 in the /etc/sysctl.conf file)
If a packet reaches the end of a chain, then is the chain policy to determine the fate of the packet (DROP/ACCEPT).

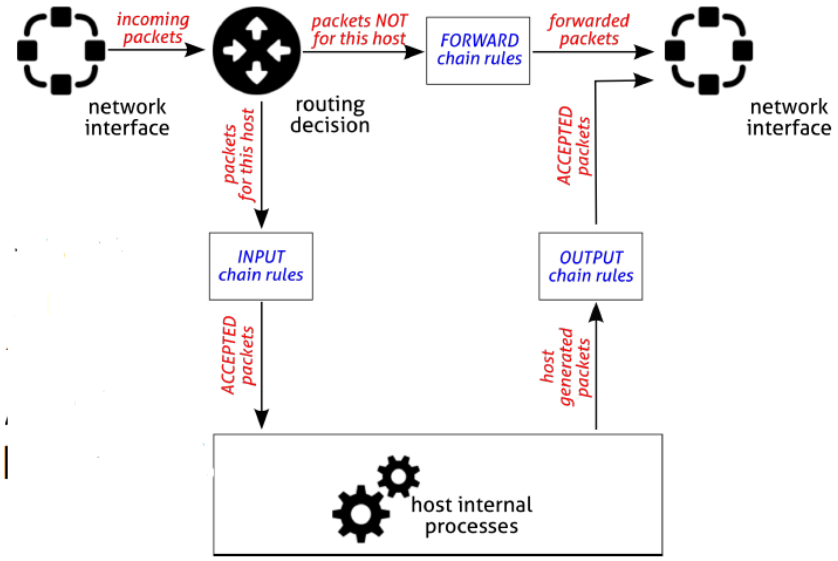


Figure 62: Packets in the Filter Table

Incoming packets are subject to routing decision: if the IP destination address of the packet is the IP address of the host itself, then the packet will be subjected to the INPUT chain rules.
If the packet is accepted, it will reach the host internal processes, otherwise it is discarded.
If the packet is not intended for the host itself, then it will be subjected to the FORWARD chain rules.
If the packet is accepted for forward, it is just forwarded, otherwise it is discarded.
The last chain is OUTPUT and it is related to packets generated by the host itself.

- Default Policy (-P parameter)

It is possible to set the default policy for a chain in this way:

- **iptables -P *chainName* [ACCEPT|DROP]**

For example: **iptables -P INPUT DROP** or **iptables -P OUTPUT ACCEPT**

- Jump Parameter (-j)

The -j parameter stands for jump to the target.
It specifies the target of the rule and what action will be performed if the packet is a match. The target could be also another chain.
These are the most frequent values:

- **ACCEPT**: accept the packet.
- **DROP**: drop the packet silently without any answer to the sender.
- **REJECT**: drop the packet and send an error message back to the sender indicating a connection failure. It is possible to specify the error message back using the *--reject-with* parameter.
- **LOG**: the packet is sent to the syslog daemon for logging.

- Protocol Parameter (-p)

The -p parameter stands for protocol. The common values are tcp, udp, icmp, all.
It is possible to say “all protocols except tcp” by using !tcp for example.
The value *all* is the default if the parameter is omitted.

TCP/UDP Protocol

It is possible to specify:

- Destination port: **--dport *portNum***
- Source port: **--sport *portNum***

It is possible to specify multiple ports by using for example:

- **-m multiport --dports 22,80,443**
- **-m multiport --sports 22,80,443**

Common services port numbers: FTP 21, SSH 22, TELNET 23, SMTP 25, DNS 53, DHCP 67-68, HTTP 80, POP3 110, NTP 123, HTTPS 443

ICMP Protocol

It is possible to specify the type of ICMP message:

- **--icmp-type *icmpType***

Type can be ***echo-reply*** or ***echo-request*** or it is possible to see different type by typing **iptables -p icmp -h**

- Source and Destination Parameter (-s and -d)

The -s and -d are used for indicating source and destination address.
They could be a network IP address (with mask /) or a plain IP Address.
In the case of a Network IP Address it means all IP addresses belonging to that network.
For example:

- **-s *192.168.1.0/24***

This statement specifies that all source IPs (-s) in the address space of 192.168.1.0 are allowed as source.
A "!" argument before the address specification inverts the sense of the address.
The analogue is for the destination parameter.

- Incoming interface (-i)

Name of an interface via which a packet was received (only for packets entering the INPUT, FORWARD and PREROUTING chains).
When the "!" argument is used before the interface name, the sense is inverted.
If the interface name ends in a "+", then any interface which begins with this name will match. If this option is omitted, any interface name will match.

- Outcoming Interface (-o)

Name of an interface via which a packet is going to be sent (for packets entering the FORWARD, OUTPUT and POSTROUTING chains).
When the "!" argument is used before the interface name, the sense is inverted. If the interface name ends in a "+", then any interface which begins with this name will match. If this option is omitted, any interface name will match.

- State (-m state --states)

This module, when combined with connection tracking, allows access to the connection tracking state for this packet, where state is a comma separated list of the connection states to match. Possible states are:

- **ESTABLISHED** meaning that the packet is associated with a connection which has seen packets in both directions
- **NEW** meaning that the packet has started a new connection, or otherwise associated with a connection which has not seen packets in both directions
- **RELATED** meaning that the packet is starting a new connection, but is associated with an existing connection, such as an FTP data transfer, or an ICMP error.

- **INVALID** the packet could not be identified

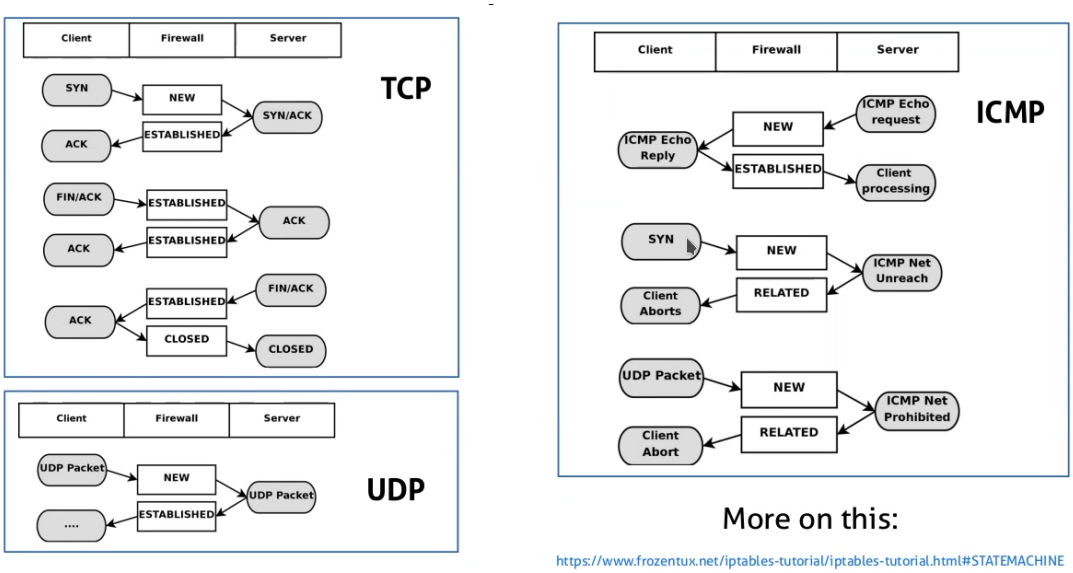


Figure 63: Connection Status with different protocols

Example:

- **-m state --state ESTABLISHED,RELATED**

- ## Examples of rules

iptables -A input -p icmp –icmp-type echo-request -j DROP
iptables -A input -p tcp –-destination-port 80 -j ACCEPT
iptables -A input -j REJECT
iptables -A FORWARD -p tcp -d 192.168.100.80 --dport 80 -s 192.168.10.2 -m state --states NEW, ESTABLISHED -j ACCEPT
iptables -A FORWARD -p tcp -s 192.168.100.80 --sport 80 -d 192.168.10.2 -m state --states ESTABLISHED -j ACCEPT
iptables -A FORWARD -s 0/0 -i eth0 -d 192.168.1.58 -o eth1 -p TCP --sport 1024:65535 -m multiport --dport 80,443 -j ACCEPT
iptables -A FORWARD -d 0/0 -o eth0 -s 192.168.1.58 -i eth1 -p TCP -m state --state ESTABLISHED -j ACCEPT

- Other useful commands

Even more useful iptables command switches

iptables switches	Description
-f	Match fragments which are not the first of a packet
--syn	Match packets which start a TCP connection It is equivalent to --tcp-flags SYN,RST,ACK SYN
-n	Used to avoid name substitution for IP hosts and ports
! (==*not*)	Used to negate the match. Can be used with many flags. Example: all the source addresses but a specific one **-s ! ip_address**
-m conntrack -ctstate...	Enable connection tracking (superset of state)
-m mac --mac-source	Match packets with a specific source MAC address
-m limit	Limit the number of packets for a specific time period --limit and --limit-burst **iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT**

Figure 64: iptables commands switches

- NAT Table

Used for NAT (Network Address Translation), so to translate the packet's source field or destination field.
Only the first packet in a stream will hit this table (the rest of the packets will automatically have the same action).
It is used to manipulate the IP Source and IP Destination, but also Source Port and Destination Port.

For the *jump parameter (-j)*, there are special targets:

- **DNAT**: destination nat, used typically when there is a server in an internal LAN that we want to make accessible from the outside. Removing, from the incoming packets, the public IP address from the destination and substitute it using the private IP address of the server.
- **SNAT**: source nat, used for outgoing packets from the internal LAN to the outside, removing the private IP address from the source and substitute it using the public IP address of the NAT device.

- **MASQUERADE**: dynamic nat, this is used when the public interface address of the firewall is dynamically assigned at that time.
- **REDIRECT**: redirects the packet to the machine itself

PREROUTING is a chain that is evaluated before the routing decisions are taken; to change something on the packet before the routing decision is take.
POSTROUTING is a chain that is evaluated after the routing decision is taken; when the packet is supposed to leave the interface.

The DNAT is used in the PREROUTING chain.
The SNAT is used in the POSTROUTING chain.
The MASQUERADE follows the SNAT.

- Mangle Table

This table is used to manipulate IP header (like ToS, TTL) and bits in TCP Header. Should not be used for FILTERING and NAT.
This table has 5 chains: PREROUTING, INPUT, OUTPUT, FORWARD, POSTROUTING.

- Raw Table

- Chain and table priorities

In every case, the priority of the table is: **1) MANGLE 2) NAT 3) FILTER**
If exits, **RAW** table has always priority on all the others.

When a packet is received from the network, it goes first through the PREROUTING chain following the priority order.
Then the routing decision is taken.
According to the destination, it will follow the chain for the FORWARD or for the INPUT.
If a packet is generated by the host, the first chain that must be considered is the OUTPUT chain and then the POSTROUTING chain.

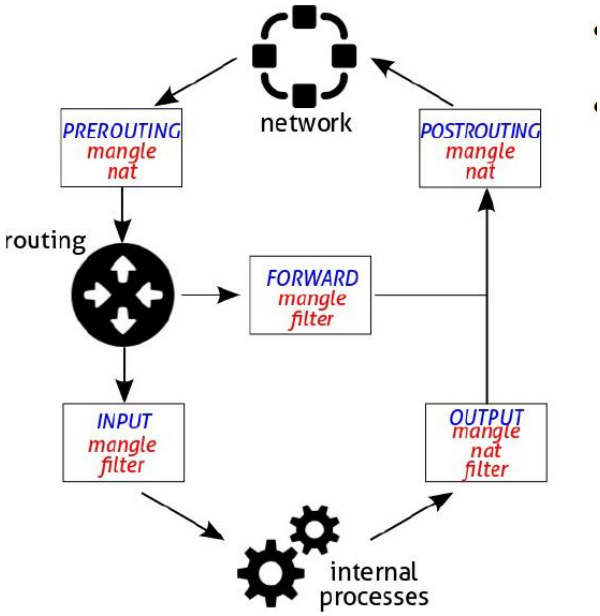


Figure 65: Tables and chains prioriies

It is possible to define additional chains the tables, called **used defined chains**.
It is possible to specify a jump action (-j parameter) to a different chain within the same table. The new chain must be user specified.
If the end of the user specified chain is reached (no matching rule), then it returns back to the invoking chain and the remaining rules are evaluated.

VIII. Network Address Translation NAT

- Routable IP Addressing

We mention that at a given point in the history of Internet, it was realized that the number of IPv4 addresses was going to be terminated very soon.
The idea to solve this problem was to introduce a distinction between public IP addresses and private IP addresses.
In this way, companies were able to use internally any kind of addressing mechanism, so that if there were the need to go outside on the Internet, they use a public IP address.
This made a distinction between **routable** and **non-routable** IP addresses.
Routable addresses need to be unique on the Internet to be PUBLICLY reachable → public IP addresses

The Non-routable addresses are called **Special-Purpose Address** IP addresses and the most common are:

- **Private addresses:** 10.0.0.0 – 10.255.255.255 (/8 prefix),
  172.16.0.0 – 172.31.255.255 (/12 prefix),
  192.168.0.0 – 192.168.255.255 (/16 prefix)
- **Loopback addresses:** 127.0.0.0 – 127.255.255.255 (/8 prefix)
- **Shared address space:** 100.64.0.0 – 100.127.255.255 (/10 prefix)

The idea is to combine the private addresses mechanism with the NAT.
We can give any kind of private addressing inside the internal network, and when it is needed to go outside the internal network, we have to substitute the private IP address with a public IP address.
For example a company may have 256 IP public addresses but these addresses could be used by thousands of hosts that can share the same IP public address playing with the NAT.

- NAT

There is a device, like a firewall, that:

- if packets are moving from internal private network to the Internet, it translates a private IP address to a public IP address.
- If packets are moving from the Internet to the internal private network, it recognizes the private IP address that mades the request and translates the public IP address to the private one.

Informally speaking, the NAT is connecting to the Internet a LAN using un-routable LAN addresses.

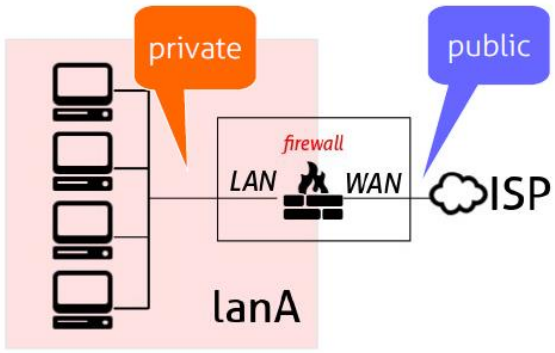


Figure 66: The NAT mechanism in a LAN

In home, the router is the only one that receives a public IP address, that is then shared by all the devices in the internal LAN.

Sometimes, there are some ISPs that also have another secondary layer of translation, and they provide an IP address that is like a second-level public IP address.
It means that during the path to the destination, the IP address could be translated more than one time. This is usually is realized with the range of the Shared-Address space, for example the IP address that the router receives by the Fastweb ISP is a non-routable IP address in the shared address space.
It is like all the customers networks of Fastweb are part of the Fastweb private network, and the public IP address used is when it is going out from the Fastweb network.

The NAT in a routed firewall:

- Can filter the requests from the Internet directed to the internal hosts;
- Allows host requests from the LAN to reach the Internet;
- All the internal LAN, all the internal hosts and its topology, is hidden to the external port scans.

The NAT goals:

- Allow private network to use just one IP address provided by ISP to connect to the Internet.
- Private networks use private IP addresses;
- Can change address of devices and the topology of the internal network without notifying outside world.
  In a public network instead, changes in routing must be notified using routing updates propagation.
- Can change ISP without changing the addresses of the devices in the private network.
- Devices inside the private LAN are not explicitly addressable by external network or visible by outside.
  This could be seen as a security plus but going back to the origins of Internet this was an issue because when it was supposed to reach and be reachable from and to any host.

The idea is that protecting the private network using NAT is something that could be done in the same and maybe better way using a well-configured firewall.
So, the debate about the implementation of NAT in IPv6 in contrast with the end-to-end full connectivity in IPv6, could be overcome using in a good way firewall instead of having NAT, because end-to-end full connectivity is not a weakness but a very useful feature of IPv6.

- ## SNAT (Source NAT)

The Source NAT translates **outgoing** IP source packet headers from the internal host addresses to the WAN IP address.
The session is **masqueraded** as coming from the NATting device.

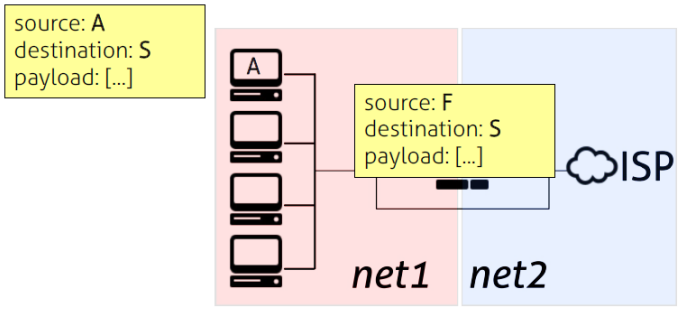


Figure 67: SNAT

The perception at the destination is that the origin of the packet is the NATing device.
Maybe there are multiple IP public addresses that could be used by the NATting device because there several networks.
In this case we call it a *Pure NAT*, there is 1-to-1 matching between one private IP address and one public IP address.
This is not what actually happens in the home networks because there are multiple devices in a single network and only 1 public IP address provided by the ISP; so we must find a way to share this unique public IP address and several different hosts in the LAN.

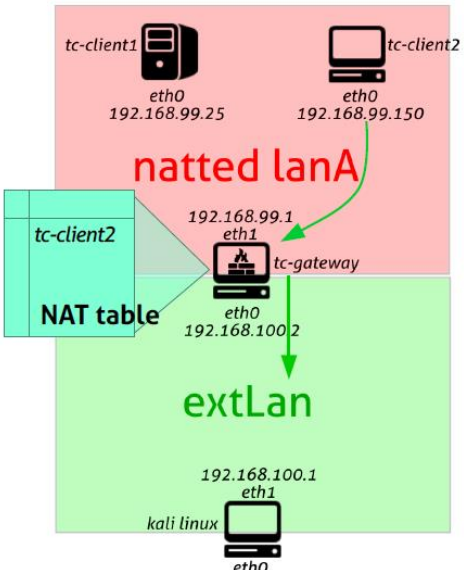


Figure 68: SNAT mechanism

The NAT mechanism operates not only at layer 3, the Network layer, but also at Transport layer. In the NATting device there is a **NAT table** that is where the associations between requests and internal IP addresses are kept.
The NATting device manages also the **related connections,** that are the other connections that are generated and opened when the real data must be exchanged; the general connection is used to exchanged commands (like get, pull, upload).

This is why it is called related connection, there is a general connection in which commands are exchanged and other connections related to this general one, that are on another port, in which the data are effectively exchanged.

- Types of NAT

The **Basic NAT** is the NAT in which there is a block of public IP addresses used for translating the addresses of the hosts inside the private LAN when they originate sessions to the external domain.
For outbound packets, from the private LAN to the Internet, the fields that are changed are the source IP address and related fields like IP, TCP, UDP and ICMP headers checksums.
For inbound packets, the fields that are changed are the destination IP address and the checksums.

The **NAPT (Network Address Port Translation)** also translates transport identifiers like TCP and UDP port or also ICMP query identifiers.
This is used to multiplex a number of private hosts into a single public IP address.
The terms NAT and NAPT are used interchangeably in literature; CISCO refers to NAPT as PAT (Port Address Translation).

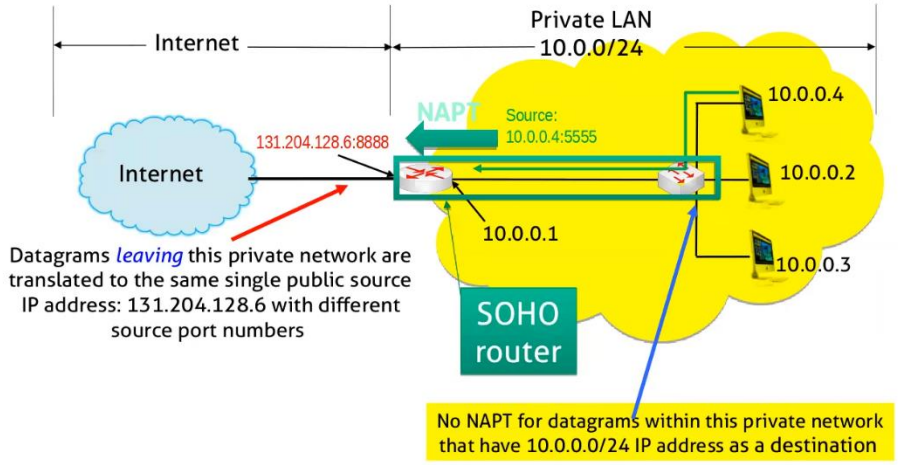


Figure 69: NAPT for outgoing packets

In the example we can see a private LAN with network address 10.0.0.0/24 and three internal hosts. We cannot really receive any packet from outside with the destinations 10.0.0.2, 10.0.0.3, 10.0.0.4 because they are non-routable addresses.
When a packet is generated from the LAN, for example by the host 10.0.0.4 with source port 5555, the router changes, for every packet it receives from the LAN, the source IP address with the public one (131.204.128.6) but the number of the port is used as an identifier actually to distinguish different requests and different sources.
The NAPT router inserts in the ***NAPT translation table*** the association between

private IP address and source port with public IP address and new port (8888 in the example).

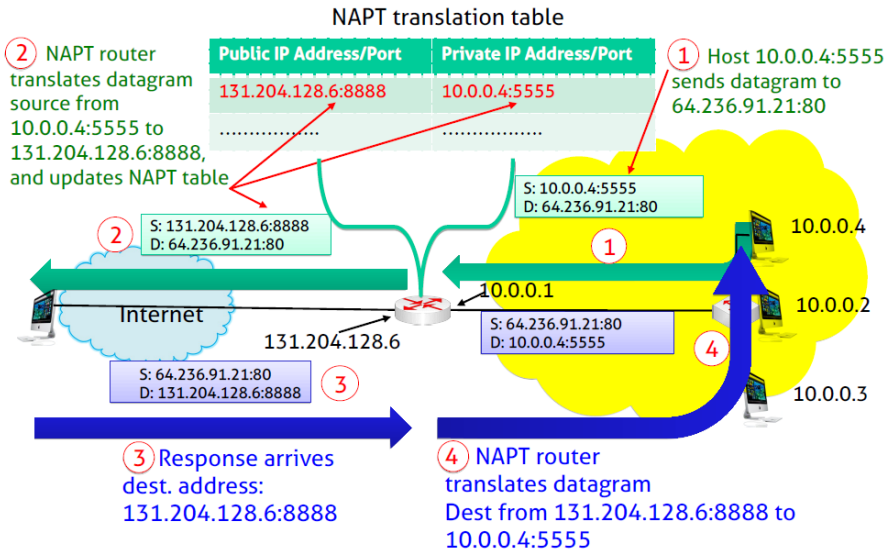


Figure 70: NAPT mechanism, ingoing packets

When the packets leave the router, it has as source IP address the public one (131.204.128.6) and source port 8888.
The server receives the request, and it appears as coming from the router.
The server will answer then using the public IP address (131.204.128.6) and the same port 8888.
When the router receives this answer, it will look on the NAPT table the association between private request and its translation.
In this way it can change the destination IP address to the private host one (10.0.0.4) and also to change the destination port from 8888 to 5555.

If there is no match in the NAPT table, the incoming requests from the Internet are dropped. So, the only replies that can pass, are the ones that already have an entry in the NAPT table. It is like a default deny rule.
But obviously this is a limitation if in the private LAN there is a server that we want to expose to the Internet.
There are different methods to overcome this problem: the most known one is the **static port forwarding**.
The idea under the port forwarding is to select some fixed ports on the router, and decide that all the ingoing packets that have as destination port these specific ports should be forwarded to a specific server inside the LAN.

There are also other solutions to make possible for an internal host to receive incoming requests like an **Application Level Gateways** (like a proxy)**, Universal Plug and Play (UPnP), Traversal Using Relays around NAT (TURN).**

- Destination NAT (DNAT)

Without any specification, the router that receives a packet intended for itself and the IP address of the destination is changed with a private IP address of an internal server and forwarded to it.
Enables servers located inside the firewall/router LAN to accessed by clients located outside. The service appears to be hosted by the firewall/router.
The IP address that is used by the client as destination address to contact the server is the public IP Address of the router.
The firewall/router uses the NAT table to:
– Translate incoming packets from the firewall/router WAN IP address to the internal address of the server
– Forward the replies to the client service request from the internal server to the external client
– Enables the client-server session or connection to continue on another port as requested by the internal server, forwarding any responses by the client to the server (the RELATED connections)

It is also called **port forwarding on Virtual Server**.
You can enable the reception of different ports on the firewalls, and according to the port accessed from the external interface, the packets could be forwarded to different internal hosts.

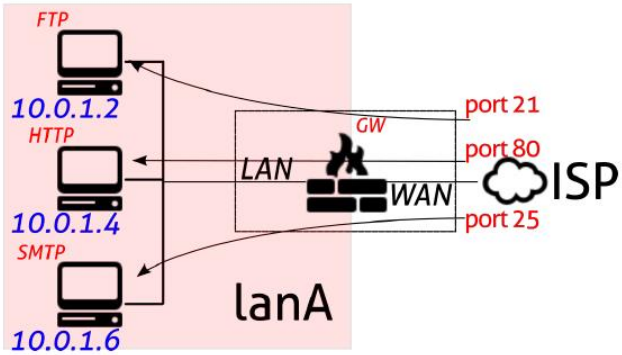


Figure 71: DNAT mechanism

This is a careful operation because we expose a device to the entire Internet, and we need to protect it using the *authentication of the request*.

When we distinguish between SNAT and DNAT it only refers to the initial transmission: to allow this mechanism we always have to translate source and destination.
If a host that is sending a packet from the internal LAN to the outside, the source IP address is translated; when the server replies then it is the destination address that is translated.

What makes the distinction between SNAT and DNAT is the first address translation.

- NAT pro and cons

– The need to establish state before anything gets through from outside to inside solves one set of problems
– The expiration of state to stop receiving any packets when finished with a flow solves a set of problems
– The ability for nodes to appear to be attached at the edge of the network solves a set of problems.
– The ability to have addresses that are not publicly routed solves yet another set (mostly changes where the state is and scale requirements for the first one).

With *RFC 4864*, the perceived benefits of NAT and impact on IPv4:

- **Simple gateway between Internet and private network**; the choke point if the traffic does not go enter through the router, there is no other way to enter to the network.
- **Simple security due to stateful filter implementation**: because the concept of state of the connection is needed to realize the mechanism, and there is the NAT Table where there is the mapping between the request with private IP address and response with public IP address.
- **User/Application Tracking**
- **Privacy and Topology Hiding:** using every time the public IP address an external host does not know anything about the internal hosts of the network.
- **Independent control of addressing in the private network**
- **Global Address Pool Conservation**
- **Multihoming and renumbering with NAT:** you can change all the private IP address of the LAN without changing the public IP Address or otherwise you can change the public IP address and maintain the private addressing of the network.

In the same RFC 4864, there's how the same previous benefits are obtained in the protocol IPv6 without using the NAT mechanism:

- **Simple gateway**: realized using DHCP Prefix Delegation in IPv6
- **Simple security**: realized using a firewall using the ACL mechanism
- **User/Application Tracking:** realized using the address uniqueness
- **End-System privacy:** realized using the temporary addresses
- **Topology Hiding:** untraceable addresses using IGP host routes
- **Global Address Pool Conservation:** not needed because there is an huge number of addresses

- **Multihoming and renumbering :** there are many IPv6 addresses for interface

Then, there are many applications that are not working with NAT, so these are the cons. Typically these are not really problems that cannot be solved but that needs tools and other efforts:

1. **Applications that have real-specific IP address information in the payload**: the translation in the NAT mechanism happens in the header, not in the payload. There will be a mismatch because there will be an address translation in the header and not in the payload.
   For this problem we can write applications that do not use IP addresses in the payload, or to use NAT with the use of Application Layer Gateways: the realization of an application that knows the use of NAT, and when it is generating the information for the payload, it performs the IP/port translation also inside it.
   It is possible to use also Interactive Connectivity Establishment ICE by using STUN servers and TURN servers.
2. **Bundled session applications**
3. **Peer-to-peer applications:** it is not possible to have a direct connection, this problem can be solved but needs specific features and tools.
4. **IP fragmentation with NAPT**
5. **Applications requiring retention of address mapping**
6. **Applications requiring more public addresses than available**
7. **Encrypted protocols like IPsec, IKE, Kerberos:** they need special changes in order to work under the NAT mechanism.

The **hole punching** is a method for establishing bidirectional connections between Internet hosts, both placed in private LANs behind firewalls using NAT.
The working depends on the NAT implementations details.
The idea under the hole punching is to find the public IP address of the other peer and you make aware the host of this creating an entry for the connection in the NAT Table, so that replies can be correctly passed.
This is called the **STUN, Sessions Traversal Utilities for NAT**, is used for this purpose, to discover and help two peers to communicate.

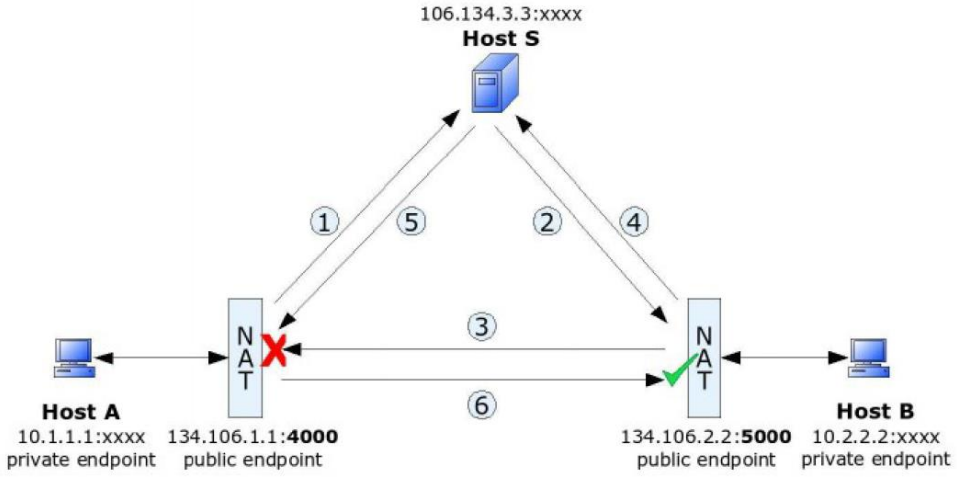


Figure 72: Example of third party for NAT traversal

Both of the Host A (10.1.1.1) and Host B (10.2.2.2) are behind a NAT device, and the NAT device for host A has the public IP address 134.106.1.1 and the NAT device for host B has the public IP address 134.160.2.2.
We need a third-party to realize this, that is a server (Host S) used only for realizing this mechanism.
1) Host A sends a request, translated using as source IP the public IP of the NAT device (134.106.1.1), to Host S saying that wants to connect to Host B behind the NAT with public IP 134.106.2.2, using the source port 4000.
2) Host S tells to the Host B that there's a request to establish a connection from the network under the 134.106.1.1 public IP with the port 4000.
3) Host B try to start a connection using the public IP address of the NAT device of Host A, 134.160.1.1, with source port 5000 and destination port 4000.
But this connection will fail because there was no connection status stored in the NAT table of the router.
4) Host B communicates to the server Host S that is awaiting for connections coming from Host A
5) Host S communicates to Host A an important information that is the port (5000 in the example) used by NAT 134.106.2.2 to start the connection.
6) Then, this port is used with the public IP of NAT B to start another connection.
In this case the connection is established, because this connection is stored in the NAT Table of router B.

IX. Link-Local Attacks

- Network eavesdropping

Network eavesdropping or network sniffing is a practice when the traffic (packets) sent by other networks is captured and the data content inside is read in search of sensitive information like passwords, session tokens, and so on.
The very first solution in order to avoid these kinds of problems is by using the standard approach of cryptography: this means that if we encrypt our data before sending them, we are safe.
But there are so many protocols that does not use encryption like DNS, HTTP, FTP and so on.
The network eavesdropping is done by using tools called network sniffers or protocol analyzers like Ettercap, bettercap, networkminer, driftnet, dsniff.
They can work in passive mode that means the packets are simply captured copied and passed at user level for further analysis. The important thing is that network eavesdropping requires to be along the path or a broadcast domain.

Like in monitoring, we have to use networking interface in promiscuous mode.
Sniffers must be along the path or at least in the same network.

- **Non switched LAN (LAN with HUBS)**: the ideal case because the hub duplicates every frame to all ports. This is exactly the virtual environment of Kathara, where it is used virtual interfaces and a bridge.
- **LAN with switches**: segmented network, in this case the switch has many ports and usually every single host is connected to one single port.
  The switch learns the MAC Addresses of all the hosts connected to the port and makes a bound between port number and MAC Address.
  In this way whenever the switch receives a packet with a specific MAC Address it knows the specific port to which forward the packet.
  In this case an attacker could:
  1) break the switch segmentation, sometimes by flooding the switch with a large number of frames that is called *MAC flooding*.
  2) performing ARP spoof attack to redirect the traffic from one port to another instead of the real destination, this means to realize the Man in the Middle attack. This attack could be realized using ARP poisoning.
- **Wireless LAN**: it is only possible if encryption is not used, or weak encryption is used. In this way the scenario becomes similar to LAN with HUBs.

Bridges was the first way to reduce collisions and segment a network.
It is nothing more than a device that connects two ports joining to network segments; only frames supposed to go to the other segment of the network are replicated. The bridge was listening all the source MAC Address on both the endpoints, if the packet was supposed to go to the other side, than replicate it (store and forward).
Switches are multiport bridges. Regenerate a frame only in the segment where the MAC Address of the destination is linked.
It learns the host in each network segment in real time,

MAC Address/CAM Table Review

CAM Table stands for Content Addressable Memory.
It stores information such as MAC addresses available on physical ports with their associated VLAN parameters.
The problem is that CAM Tables have a *fixed size*, because they have to be extremely fast and efficient.
As frames move in the switches, the CAM is filled with the MAC Addresses.
If a MAC Address is not anymore in the memory or is unknown, the packet is replicated on all the possible ports, causing a flooding.

CAM overflow

This is a theoretical attack until May 1999 and it is based on CAM Table's limited size. Usually switches use hash to place MAC Address in CAM Table like hashed lists. If all the entries in the buckets are filled, then the packet is flooded.
The attack is based, as seen, on the CAM Table's limited size.
Now what happens depends on the switch implementation:

- Switch starts flooding and then the attack has success.
- Switch freezes or crashes (DoS scenario)

Today this kind of attack is not effective because of port security in switches.
The idea is that is possible to specify MAC Address for each port or to learn a certain amount of MAC Addresses per port; for example, if a switch is connected to another switch, the port that is used to connect both there will be all the MAC Addresses of the 2nd switch. Upon a detection of an invalid MAC the switch can be configured to block only the offending MAC Address or to block the port.

- ## ARP Spoofing

In this case, we want to foul one of the hosts or several hosts to send packets to the attacker instead of the real destination.
An ARP (Address Resolution Protocol) **ARP-request** message should be placed in an Ethernet frame and broadcast to all computers on the network; the host A known an IP Address that is in the same network and then knows that it can have the MAC Address of the device that is in the same network.
So the message request will have as source the sender's MAC Address and IP Address, as MAC destination all the possible MAC Address (FF:FF:FF:FF:FF:FF) and IP destination the IP Address.
Each computer receives the requests and examines the IP Address.
The computer that has the IP Address mentioned in the request, host B, sends back an **ARP-reply** response with source address its MAC address and destination address the host A MAC Address.
All other hosts simply discard the request without sending a response.

This information, the association MAC Address-IP Address of every host are stored in a dynamic table called **ARP Table** (in IPv6 it is called *Neighbor Table*).
It is accessed before sending any Ethernet Frame, if the MAC is already there is not necessary to send an ARP request.
It starts empty and is filled as the MAC Addresses are collected.
Unused MAC addresses are removed after a timeout in order of minutes.
According to RFC 826 (ARP), when receiving an ARP reply, the IP-MAC pairing is updated.
Add to this, there is the **Gratuitous ARP response**.
It is used by hosts as an announcement containing their IP address to the local network and to avoid duplicate IP addresses on the network (like the DAD Duplicate Address Detection in IPv6).
Routers and network hardware may use cache information gained from gratuitous ARP responses. Gratuitous ARP is a broadcast packet, like ARP request.

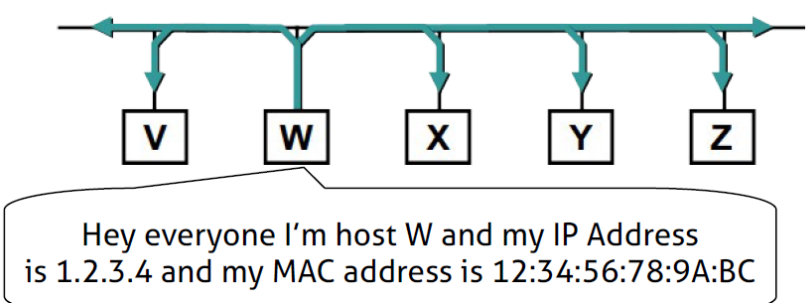


Figure 73: Gratuitous ARP response example

The Gratuitous ARP message is something like: “Hi, I’m host W and my IP Address is 1.2.3.4 and my MAC Address is 12:34:56:78:9A:BC”.
The problem of this is **that ARP has no kind of security of ownership of IP Address or MAC Address**; so if host W wants to take the identity of the gateway can simply send an gratuitous ARP message and saying that host W has IP 1.2.3.1 and MAC 12:34:56:78:9A:BC.
So other hosts *will use the correct IP Address of the gateway with the attacker MAC Address.*
With an ARP spoofing an attacker could pretend to be anybody like one of the host, the default gateway, the DNS, and so on; so the attacker sends the attacker MAC address with the IP address of another device, in this way the ARP Table of the hosts is updated with these fake associations.
In this way it is possible to launch:

- a **DoS attack**, because the real host will never receive packets intended for it.
- a **Man in the Middle Attack**: attacker intercepts the traffic, reroute it to get the reply, then forward the reply back and in the meanwhile examine or sniff or alter the data.

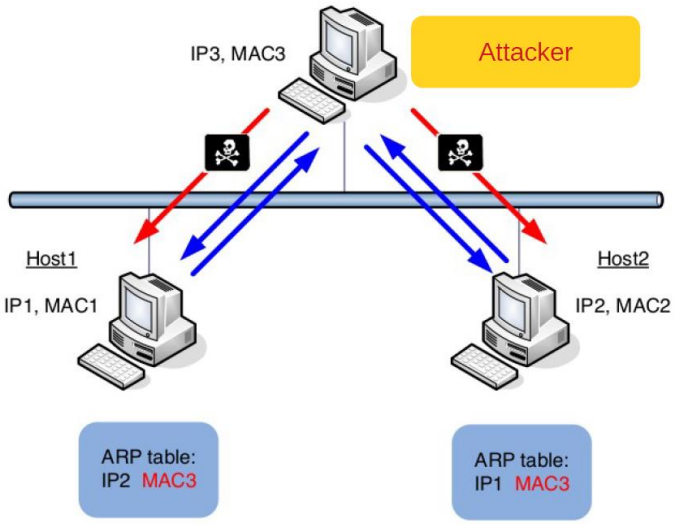


Figure 74: Example of MITM attack, working like a proxy

- IPv6 Neighbor Discovery Threats

IPv6 Neighbor Discovery is used like the ARP protocol in IPv4.
The Neighbor Discovery uses the **Solicited Node Multicast** in IPv6 and Multicast at Ethernet level. The messages involved in this mechanism are the Neighbor Solicitation and the Neighbor Advertisement.
DAD is used to guarantee that an IPv6 unicast address is unique on the same link. A device will send a Neighbor Solicitation for its own unicast address, static or dynamic.
After a period of time, if a NA is not received the address is deemed to be unique. Then it is possible to make a **DoS Denial of Service** attack if an attacker host always reply with a Neighbor Advertisement to the host that is asking if there is another host with its IP Address in the link making impossible for him to connect to the network.

Another kind of attack is the ***ICMPv6 Redirect attack***.
In a network there are several routers that are accessible, a router informs an originating host of the IP address of a router that is on the local link and is closer to the destination. For example, in the figure below, PCA has as default gateway R1 and wants to reach network X; so R1 sends a packet to PCA by telling that there is another router, R2, that is closer to network X if PCA wants to reach that network. In this way PCA can add to its routing table a new route to reach network X passing through R2.
Another ICMP redirect mechanism because of IPv6 is that if PCA wants to reach PCB, this is tricky because in IPv6 there could more network prefixes and two hosts with different network prefixes can share the same link, then the router R1 says to PCA that they are on the same link and could forward directly the packet to PCB.

The vulnerabilities related to these mechanisms of ICMPv6 redirects and Neighbor Discovery protocol then we have these threats:

- **Routing threats**: malicious last hop router, default router compromise, spoof redirect messages, bogus on-link prefix, bogus address configuration prefix, parameter spoofing.
- **Non router threats**: neighbor NA/NS spoofing, neighbor unreachability Detection (NUD) failure, duplicate address detection DoS attack
- **Replay Attacks**
- **Neighbor Discovery DoS Attacks**

Rogue RA

The VPN bypass, with a special RA advertisement, it is possible to convince hosts to user another route for the VPN packets or to not use the VPN. It is called tunnel split because you do not enter the tunnel but go in a parallel way.

RA flooding

Flooding IPv6 hosts with Router Advertisements. The host that receives a RA, it has to create a new IP Address, to evaluate new routes.
In previous implementation the result of RA flooding is a system crash and DoS.

DHCP rogue server or DHCP starvation

If there is a DHCP server, the attacker can perform a DHCP starvation by pretend to be many hosts by requesting all of the available DHCP addresses.
Once all the addresses are gone, the attacker could use a rogue DHCP server to provide itself the addresses to the hosts.
Since DHCP response include DNS server and default gateway address, the attacker can pretend to be any of these entities.

X. Network hardening

- Introduction

Network hardening means the protection of the network devices.
There are many different types of devices, and each of them must be protected.
If even one of them is breached, then there could be some risks that can extend to the entire network or infrastructure that could be compromised.
The idea is to provide a pragmatic approach, like following a manual with some guidelines, the use of methodology to protect network devices makes it possible to reduce the risk of violations and to limit the impact of anomalous events, whether they are voluntary (attacks) or involuntary (human errors or failures).

- Scopes of action

Management plane
The scope of the network device management.
It consists of an administrator's protocols and tools to configure, change configurations or parameters, monitor or access a network device (e.g. SSH, SNMP, NTP).
This is when you access the configuration panel of the devices.
Breaches in this area can be caused by simple passwords or insecure protocols, resulting in unauthorized access or loss of access to the device.

Control plane
The scope of the support for the operation of network devices; it must not be confused with management plane, and it is the level of behavior of the devices that use for regular operations.
For example, the routing messages that a router receives, the SNMP messages that a host receives, the ICMP packets, alter database entry, all packets intended for the network devices (Router Advertisements and Router updates) and so on.
It consists of the protocols and mechanisms devices used to perform their tasks; the control plane can alter the information that device uses for regular operation.
Violations in this area are usually caused by unauthorized data exchange with device, resulting in a loss of performance (Denial of Service).
The control plane is made of packets that are intended for the devices like Router Advertisements and Router updates.

Data plane

The scope of operation of network devices.
It corresponds to traffic forwarded by network devices (switches, routers, firewalls) and the paths that appliances choose for individual packets.
The data that must be forwarded not intended for the network devices.
Violations in this area caused by external events (congestions), malicious attacks (spoofing, redirect, hijacking, and so on), and failures can result in the alteration of packet paths and a block of networks services.
Protection must occur at all scopes. In all three areas of action of the network devices it is possible to have violations; the three areas are closely linked.
It is important to adopt best practices to protect devices in all three areas.

Examples of problems in the homework lab:

- Default weak password used in Management plane.
- Use the HTTP protocol to access the firewall that is not a secure protocol in Management Plane.
- Firewall Rules and Policy is related to the Data Plane.

- Management Plane

Appliance access protection

Access to a device is the first step in configuring its operation an determining its behavior.
There are two kinds of access:

- *physical access to the device*: with a serial terminal or a laptop connected to the console port. The risk in this case is reduced since the access is restricted physically, you can access only by entering in the room where the device is placed.
- *remote access to the device*: through the network itself or using a virtual terminal such ssh or telnet. Then, in that case the risk is higher since anyone who can send traffic that reaches the device can claim to be a network admin.

Access to a device allows access to other network devices monitoring and status management functions, usually through the SNMP protocol.

Password policy

Passwords are the simplest and most widely used form of authentication.
It is good to use password that are hard to guess, for example:
- password at least 8 characters long;

- combinations of alphanumeric, upper and lower case characters, punctuation marks;
- passwords that are changed frequently;
- passwords not too complex because they are really hard to remember.

Brute force protection

The use of different character types and a minimum length ensures that the passwords search space is enormous so that a brute force attack is hard to accomplish.

It is good to configure devices that they can store passwords in an encrypted way and temporary block accounts if the passwords is wrong three times in a little amount of time, notifying the incident in the logs.

AAA Principle

AAA is the principle of reducing unauthorized access to resources.

The idea is to identify users that are accessing to devices and resources, check that they have permissions only on those components for which they are authorized and finally keep a record of their actions to reconstruct the chain of events.

- *Authentication*: verifying the identity of user by username and password, or token or similar.
- *Authorization*: verify if a user is authorized to act on a system; the association between users and permission can be done in different ways, the best one is the RBAC, Role-Based Access Control which considers the presence of different roles in the system or group of users with same permissions. In RBAC, each users is associated with one or more role, and each role has a set of permissions.
- *Accounting/Auditing*: store the details (time, duration, command used, author) of each action taken by each user in a permanent, unalterable log. Usually network devices can send information to different destination like terminal, SNMP server, Syslog server.

Use centralized solutions

The idea is that whenever we are using the principle of the AAA, it is much less cumbersome to have one single place where is it possible to view users and associate permissions on the system.

This allows you to have centralized point from which to manage all the devices, users, permissions on the network.

These functions use specific protocols for managing users, passwords, permissions, authorization and authentication checks, and so on.

The alternative to centralized solution is the local solution, where each device has its own set of users, passwords and permissions.
In this way a single user can potentially have different accounts in different devices with different passwords and permissions; so even if it seems a simpler solution it can be complex to manage.

Use a reliable NTP server
It is important that there is the reliability of time information: all the devices in the network are synchronized with the time. If there is some delay, there could be some misalignments in the time; this is critical when understanding and reconstructing the series of events when they involve several different devices.
For this purpose, the *NTP Network Time Protocol* is used.
Usually, one of the devices on the network is designated as the reference NTP server, and all other devices interact with it to synchronize with its time.
This device in turn can refer to another trusted NTP server, to receive the current time accurately.
This ensures that all logs from all devices can be compared temporally.

Syslog
Syslog is standard mechanism for generating log messages.
Its structure allows you to efficiently separate the applications that generate logs from those that need to store them and those that need to consult them.
The syslog server is critical to centrally collecting and managing logs.

Example:
Timestamp – Facility – Severity – Mnemonic - Message
AUG 31 14:40:10: %LINK-5-CHANGED: Interface Fast Ethernet changed state to down.

Syslog divide types of events in eight levels of criticality called Severity for individual log messages in an increasing priority, ordered in such a way that levels with low values have higher criticality than those with higher values:
0 Emergencies, 1 Alert, 2 Critical, 3 Errors, 4 Warnings, 5 Notification, 6 Informational, 7 Debugging.

Devices send log messages with a level less than or equal to the set criticality level: this implies that that if for example the set criticality at Warning Level the only messages that will be sent are from priority 4 to 0.
If the level of criticality is high, then the number of possible messages will increase with more groups of priorities.

Is therefore essential to choose appropriate level of criticality (for example set the use of Debugging only to the time required to solve an anomaly or while configuring) and to be sure that the syslog server has an adequate log storage capacity, both in terms of space and computing capacity.

Remotely access using encryption
Remotely accessing network devices must be done in a secure way by using encryption protocols like HTTPS and avoid the use of HTTP protocol.
For devices that require command-line configuration (CLI) avoid telnet that is not encrypted protocol, but try to use a secure protocol like SSH. Or to use telnet in an isolated channel or within an encrypted channel (like a VPN).
SSH should always be preferred as it offers the same functionality of telnet but encrypts every packet exchanged between the device and the administrator.

- Control Plane

Control Plane is the use of the data that devices need to work; they use some kind of information received by other devices for their daily activities.
The protection of control data is done by avoiding unauthorized changes to the way traffic moves through the network and avoid overloading devices (DoS).
For this purpose, we use:
- Control Plane Policing and Control Plane Protection, measures that can limit packets addressed directly to the devices and that require the use of their CPUs. This is related to the protection of DoS.
- Routing protocols with authentication reduce the risk of using information for non-genuine traffic routing.

DoS protection
The primary purpose of routers is to forward packets; so, routers are very optimized. Packet forwarding is almost always done from the cache, without using too much the CPU so there is zero effort in this operation.
On the other hand process packets directed to a router, involve the CPU, possibly in a considerable way: for examples, routing table updates, management traffic (telnet, ssh, SNMP), service traffic like DHCP, IP options that require processing by routers (data fragmentation, extensions header in IPv6 hop-by-hop, and so on).

Therefore, flooding routers with packers for processing can impact their performance and cause a DoS attack.
The specific protection for the control plane is to limit this kind of traffic, setting up thresholds for the reception: for example, x packets for second for protocol y, z traffic only from interface k, ignore protocol w, and so on.

Manage dangerous ICMP packets

The ICMP protocol can also have adverse effects on the CPUs of network devices. Some kind of ICMP packets like the ICMP redirect are dangerous, so we need to avoid these kinds of packets.

Considering that the ICMP packets are informative, some types of packets can be disabled in the devices without altering the functionality of the network; these packets are intended to be used when a network topology changes, then if the network does not change, this kind of traffic is not needed.

To limit the impact on the CPU of network devices, it is a good practice to use the ICMP packet filtering mechanism to block the following ICMP packets:

- **ICMP redirects**, which suggest an alternate route if the destination can be reached through another router in the same network; this can expose a MITM attack.
- **ICMP unreachable**, which informs the sender of a packet that the final destination is unavailable. This can be used to infer some information related to the internal structure of the network.

Only use authenticated routing protocols

Routing information is critical to forwarding packets according to the correct routes. It is essential to ensure that packet with routing information from other routers are authentic.

For this reason, it is best to use the authenticated version of the routing protocols; in this way false router cannot share false information.

- Data Plane

The purpose of network devices is to move packets through the network according to the security policies established by governance.

Without proper protections, attacks can be made to alter packet forwarding rules, potentially causing security policy violations.

In addition, it is challenging to observe security policy violations without proper monitoring.

Operate at all protocol stack layers

To protect the data plan, it is necessary to intervene in several protocol layers.

Level 2 protect devices from possible MAC-IP association changes.

Level 3 protect devices from IP packets with dangerous configuration that attempt to map the network.

Level 4 protect devices from ICMP packets that may alter the normal flow of packets within the network.

Level 2 Protection (Switch Configuration)

- Disable gratuitous ARP packets: to avoid ARP spoofing attacks like MAC Address theft and MITM attacks
- Enable dynamic ARP inspection (DAI) mechanism: to protect against ARP spoofing and ARP poisoning. If the switch learns the association between IP and MAC and ensures that kind of association never changes; if something change it blocks the port.
- Disable ARP Proxy IP if not needed
- Enable port security mechanism: limit the number of MAC addresses that can be linked to a single port.
- Enable DHCP snooping mechanism: this allows DHCP response packets to be forwarded only from authorized ports connected to a trusted DHCP server so it's not possible to pretend to be a DHCP server on another untrusted port.
- Enabling the IP source guard mechanism: to allow block all traffic with an abnormal IP-MAC address association, typical of IP spoofing.

Level 3 and Level 4 protection

The main tools to protect the network at layer 3 and 4 are Access Control Lists when using router or Firewall rules when we are using firewalls.

They can identify packets by considering only IP addresses (standard ACL) or IP addresses and Layer 4 header information (extended ACL). Depending on the specified policy, routers can transform packets (e.g., blocked, subjected to NAT, forwarded in a VPN, etc.).
For the protection of networks, the primary use is to filter packets (packet filtering) according to the network's security policy: they are also used for NAT, Quality of service (QoS), VPN traffic selection, policy-based routing, or routing information.

Use rules for traffic filtering: ACL in routers

- Block packets with IP address spoofing: block all packets used as source IP addresses IPs that are not consistent with the network topology, for example a host in 10.0.0.0/8 that uses IP 11.0.0.1.
- Block packets that can lead to network mapping (scanning): block functional features like to block UDP and ICMP packets coming from the outside the web that may reveal information about the internal structure.

- Allow only packets that correspond to the traffic expected on the network: according to the principle of least permissions, blocking everything that is not explicitly expected to be exchanged in the network would be a good idea.
For example, if in the DMZ there are a web server and proxy server those are the only expected traffic that you have to allow.

Use rules to authorize only trusted sources: ACL in routers
Routers can use ACLs to limit the type of traffic of supporting protocols. For example:
- an ACL allowing NTP traffic from a trusted server only;
- an ACL allowing SSH access only from administrator hosts;
- an ACL allowing ICMP or SNMP diagnostic packets from the administrator’s hosts only.

XI. Virtual Private Network (VPN)

- VPN Definition

VPN stands for Virtual Private Network, according to the definition of the NIST SP800-113, a virtual network, built on top of an existing network infrastructure, which can provide a secure communications mechanism for data and other information transferred between two endpoints.

This is a very general definition, because it doesn't say how and where you have to perform the transformation of the data, there is no strong definition of what a "secure communication" is.
According to where we decide to apply and place the encryption endpoints, we could have different combinations and types of VPN.
There could be different kinds of encryption for *Authentication, Integrity, Authenticity, Confidentiality.*
There could be different parts of communications that should be encrypted: difference between VPN performed in tunnel mode or VPN performed in transport mode.
The important goal is the usability, the solution must be easy to use, if it is hard, it will not be used.
The security goals for a VPN are:

- traditional: confidentiality of data, integrity of data, peer authentication.
- extended: replay protection, access control, traffic analysis protection.

Traffic Analysis means that even if the data are encrypted, by analyzing the traffic by doing eavesdropping, an attacker could infer some information like protocols used, IP addresses involved, and so on.
This is a typical attack that you can imagine to be applied on the Tor Network, for example to infer the kind of traffic that is generating to which kind of service is linked by previous traffic experience.

The usability goals for VPN:

- *Transparency*: VPN should be invisible to users, software and hardware. OpenVPN and IPSec are very transparent for example.
- *Flexibility*: they can adapt to different scenario, different type of hosts, applications, users. OpenVPN and IPSec are very flexible.
- *Simplicity*: VPN must be easy to be setup and usable. IPSec is a nightmare!

- Different types of VPN

Site-to-site security

In this scenario the endpoints of the VPN are the network devices (routers) that are the edge-routers of the sites. The idea is that you bind together two different sites so that any host in one site can reach the other one and the network, that the host crosses when it is on top of the VPN, will not really able to distinguish which devices are communicating.

Host-to-site security

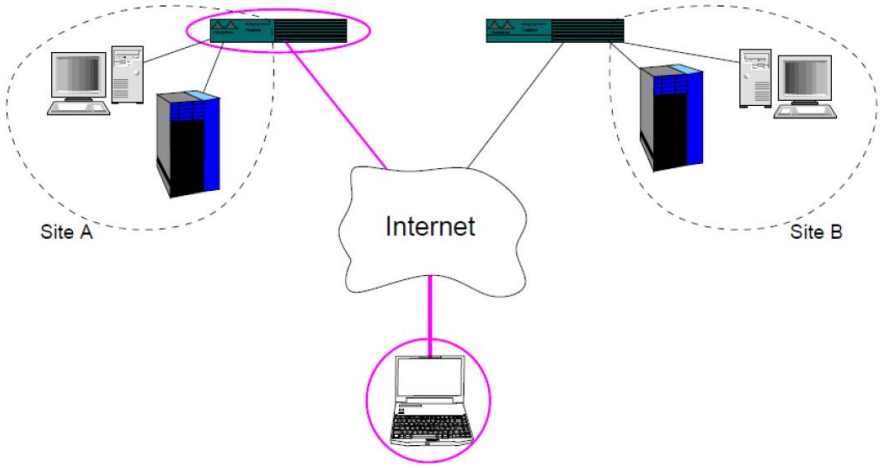


Figure 75: Host-to-site VPN

In this scenario, in the connection between the VPN endpoints, one is a network device connected to a network and the other one is a single host.
So, the idea is to extend a site to include another host, so it makes possible to access the internal site.
This is what we use for connecting to the internal VPN ACME.

Host-to-host security

In this scenario, in the connection between the VPN endpoints are two single hosts. The only encrypted connection is between the two single hosts.

- Types of VPN by layers

The encryption mechanism and the cryptography functions can be applied on all of the IP/TCP Model that are Application Layer, Transport Layer, Network Layer, Data Link Layer and Physical Layer.
By doing these operations on different layers, you obtain different effects.

Physical Layer
The physical layer is actually on the cable; at this layer we can encrypt the bits data on the cable directly. This is used by quantum cryptography in optical fiber.
Confidentiality: on the cable, Integrity: on the cable, Access control: physical access, Authentication: not needed, at the other end one single host,
Replay protection: not applied; Traffic Analysis protection: on cable;
Access Control: physical access; Transparency: full transparency;
Flexibility: hard to add new site, like special hardware dedicated for encryption.
Simplicity: excellent.

Datalink Layer
Virtual cable, like a meta-access to the data.
Confidentiality: on the link, Integrity: on the link, Access control: physical access, Authentication: not needed; Replay protection: not applied; Traffic Analysis protection: on cable; Transparency: full transparency; Flexibility: hard to add new site; Simplicity: excellent.

Network Layer
Confidentiality & Integrity: between hosts/sites;
Authentication: for host or site; Replay protection: between hosts/sites;
Traffic Analysis protection: host/site information is exposed.
Access Control: to host/site; Transparency user and SW transparency is possible;
Flexibility: can be done but needs some HW or SW modifications.
Simplicity: good for site-to-site; not good for host-to-host (every host must be configured, and it is not simple).

Transport Layer
Confidentiality: between apps/hosts/sites; Integrity: between apps/hosts/sites;
Authentication: between apps/hosts/sites; Replay protection: between aps/hosts/sites; Traffic analysis protection: protocol/host/site info, exposed; Access control: user/host/site; Transparency: user and SW transparency possible;
Flexibility: HW or SW modifications;
Simplicity: good for site to site, also good for host to site

Application Layer
Confidentiality: between users/apps Integrity: between users/apps authentication: user Replay protection: between apps
Traffic analysis protection: all but data exposed
Access control: only data access secured
Transparency: only user transparency
Flexibility: SW modifications
Simplicity: depends on application

It looks best to introduce security in the

- *Network Layer (the VPN realized using IPSec)*
- *Transport Layer (VPN realized using OpenVPN)*

These are the most popular choices of VPNs.

Other options are Secure Application layer protocols, like used for specialized purposes like S/MIME or PGP for secure e-amil.
Secure Data Link Layer protocols are most used with PPP or other point-to-point communication like PPTP and this is typically realized by the ISP that have cables to connect different sites of the organization, and usually realizes a layer-2 tunnel protocol.

• VPN Device placement

Device placement in the topology of the network is important because it affects the security, the functionality and the performance.
Main options for placement are:

- VPN functionality is in the firewall
- VPN device in the internal network
- Single-Interface VPN device in DMZ
- Dual-interface VPN device in DMZ

Remember that the cryptographic protection only extends up to the termination of the VPN device.

Firewall with an SSL VPN

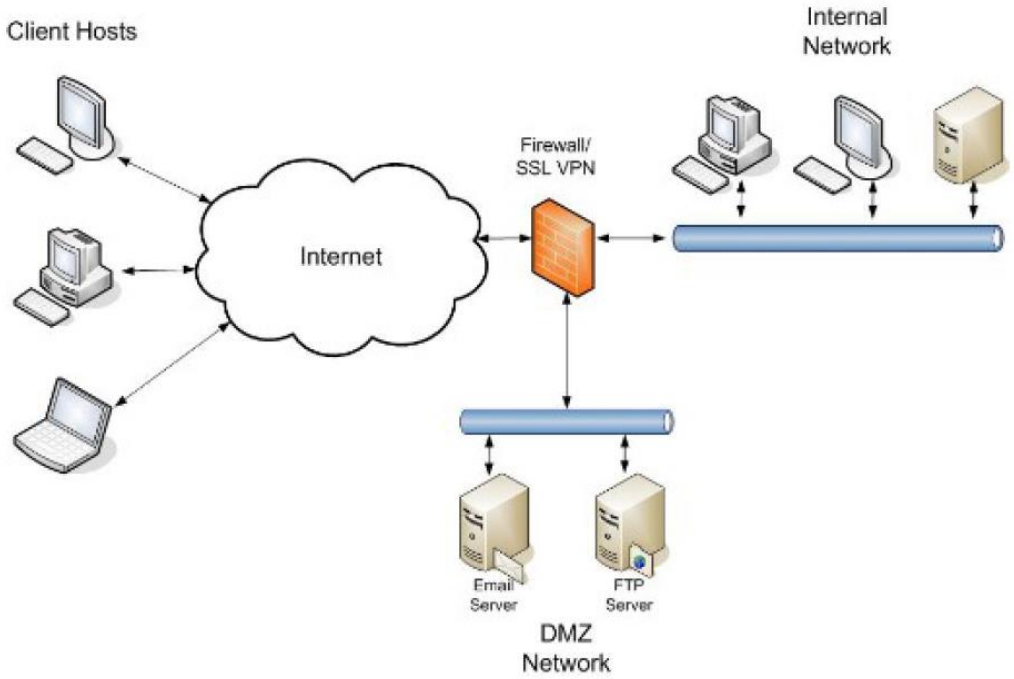


Figure 76: Firewall with an SSL VPN

The firewall is also performing one of the endpoints of the VPN tunnel.
The firewall is running a special service that is realizing a VPN; the firewall receives the traffic, removes the encryption and can perform decisions based on the traffic. This is a common solution.

Pro:
The VPN device communicates directly with internal hosts.
No holes in FW between external VPN device and internal network.
There is no needing to open special ports or setup some configuration because it's the firewall itself that receives traffic and performs the checks.
The traffic between device and internal network must go through the firewall.
Simple network administration, one single device that performs these activities.

Disadvantages:
The VPN functionality are limited to the ones that are offered by the firewall vendor.
The firewall is usually directly accessible to external users via port 443.
Adding VPN functionality to FW can introduces vulnerabilities, and it is possible to expose the entire network.
TCP Port 443 must be open on external FW interface, so that clients can initiate connections.

SSL VPN in internal network

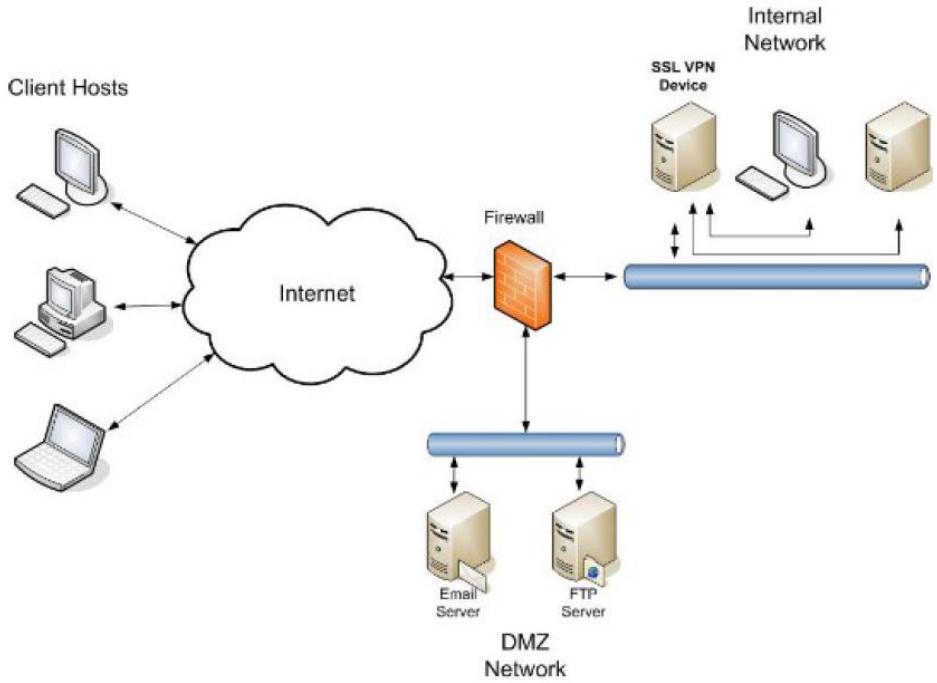


Figure 77: SSL VPN in the internal network

Whenever you have to client that wants to join the network, the traffic should cross the firewall, then the SSL VPN device process the data and decide to forward into the network or not. In some way the SSL device would be a kind of second firewall. The VPN device is placed in the internal network.

Advantages:
One single rule for single to be added to FW.
No holes needed in FW between VPN device and internal network, because it is already in the internal network.
VPN traffic is already behind the firewall, so protected from attacks by machines in DMZ.

Disadvantages:
VPN traffic passes through FW on tunnel, so it is not analyzed.
Unsolicited traffic can be sent into internal network from outside to internal VPN device.
Internal network is compromised if VPN device is compromised.

SSL VPN in DMZ with one single interface

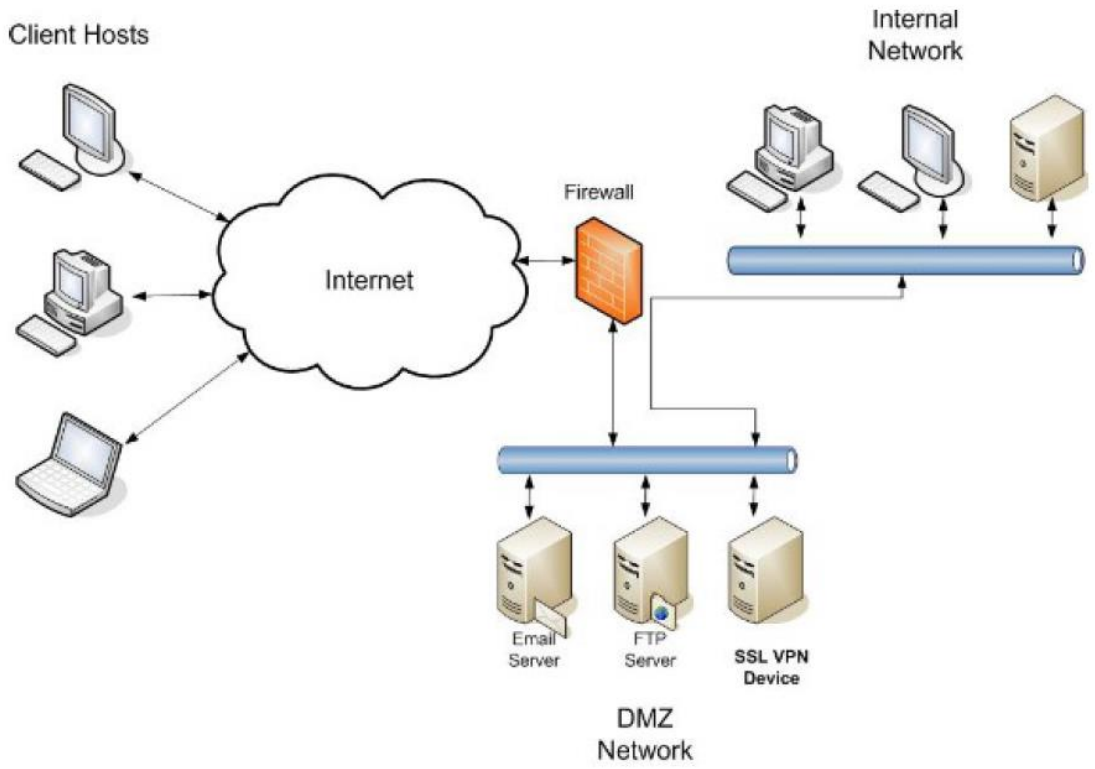


Figure 78: VPN in the DMZ network

The VPN device is in the DMZ.

Advantages:
The internal network is protected if VPN is compromised because it goes directly in the DMZ, there is no direct link between the VPN and the internal network.
Once the traffic reaches the VPN and is processed, then the data is in clear text and it is possible to use IDS (Intrusion Detection System) to analyze traffic intended for the internal network before it goes into the internal network.

Disadvantages:
- Bit more difficult to implement this connection: first it is needed to allow traffic to reach the VPN device, then it is not known which kind of traffic will be out from the VPN device, so you have also kind of traffics that we want to accept to direct to internal network. So there are potentially numerous ports to open in the FW.
- Once the traffic is decrypted from the VPN device, then is in clear text into the DMZ network, so it is unprotected. This is because the VPN device has only one single interface and it is not possible to split traffic between the two networks.
- Firewall is bypassed when user traffic is directed to the DMZ network.

Dual interfaces VPN device in DMZ

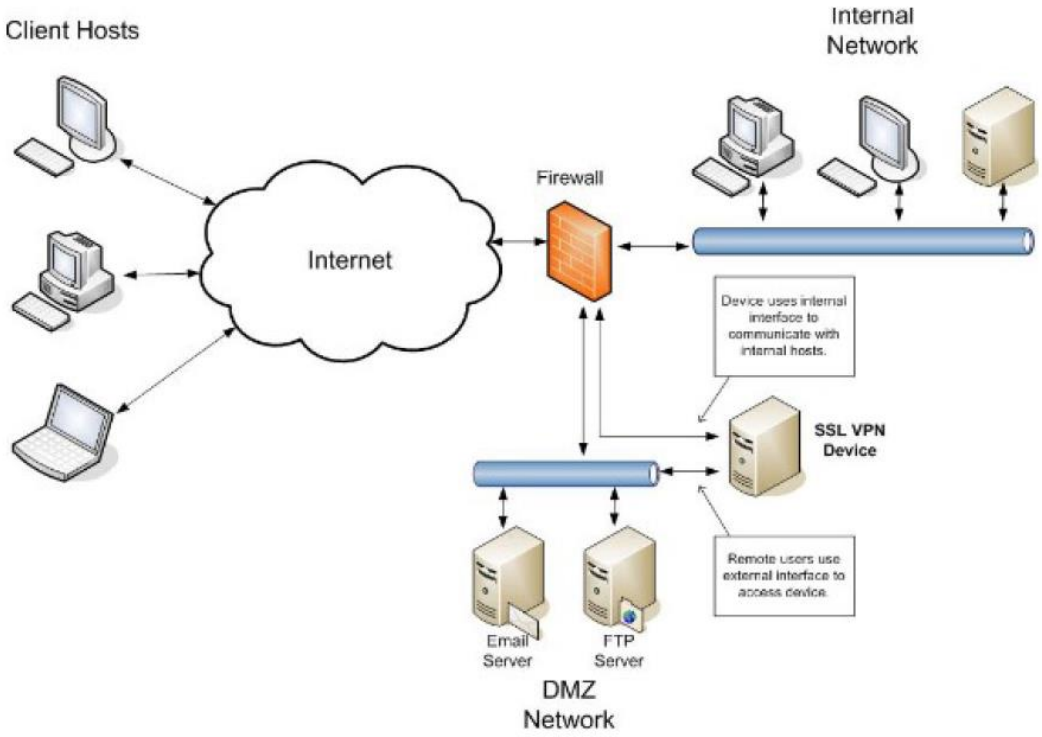


Figure 79: VPN with two interfaces in DMZ

There is a VPN device with two different interfaces, in this way the traffic that comes out the VPN device, does not enter again the DMZ network, but will follow a different link to reach the internal network.

Advantages:

- All the advantages of VPN device in DMZ.
- Unencrypted traffic to internal hosts is protected from the hosts in the DMZ.
- Only Firewall interface connected to device's internal interface needs to permit traffic from VPN device.

Disadvantages:

- Bit more difficult to implement this connection with two different interfaces.
- Introducing additional routing complexity.
- Firewall is bypassed if split tunneling is not used, and the traffic coming out from the VPN device is destinated for hosts in the DMZ network.

- Tunneling

Tunneling is the operation that performs a network connection on top of another network connection.
It allows two hosts or sites to communicate through another network that they do not want to use directly.
For example, Host on site A and Host on site B want to communicate without directly using the Internet, that is untrusted network, so they use the tunnel that is established between the two hosts.
There is an encrypted tunnel, so every packet sent from site A to site B and the other way around, will be encrypted into the tunnel.
In the Internet we are not able to see the real source and the real destination of the packets; everything appears to be only exchanged by the two sites.

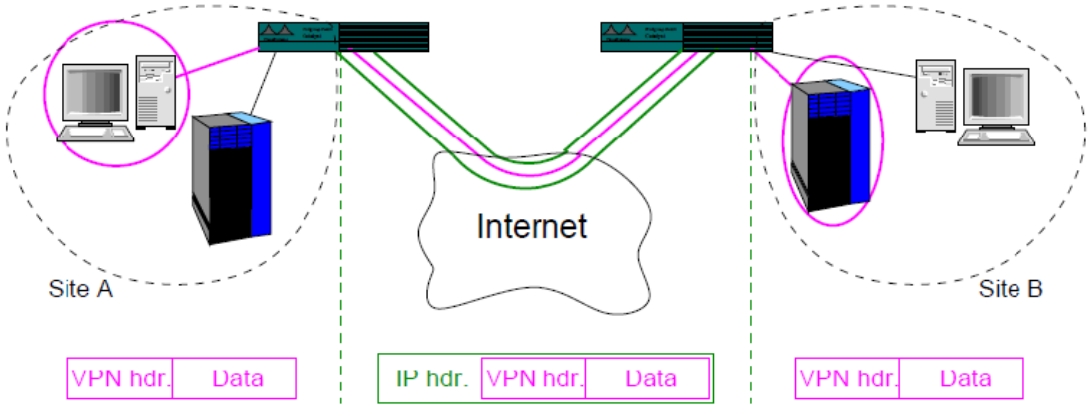


Figure 80: Site-to-site tunneling

In the **site-to-site tunneling**, it is very generic high-level transport of data, where any kind of PDU (Protocol Data Unit) will be transport on the other site of the tunnel.
The main concept is based on the fact of the encapsulation of the whole PDU in another PDU that is sent out on the network connecting the two sites.
The encapsulation is done in the edge router on the source site.
Then the routing standard mechanism of the Internet will forward the packet to the other side of the tunnel.
The decapsulation is done in the edge router on the destination site.
Note that host-to-host communication does not need the use of IP.
The VPN Header plus the Data Payload, is placed as it is, as payload inside an IP packet.
The site-to-site tunneling is a very generic approach, is not really focused in the sense of private, but to a generic exchange. For example, the join of two IPv6 network by the use of IPv4 protocol, so by having IPv6 on top of IPv4.

When encryption is introduced, the site-to-site tunneling is called **secure tunneling**.
This enables a PDU to be transported from one site to another without its contents being seen or changed by hosts on the route.
The encryption and the encapsulation is done in the edge router on the source site.
Then the routing standard mechanism of the Internet will forward the packet to the other side of the tunnel.
The decryption and decapsulation is done in the edge router on the destination site.
So, for the Internet, the two communicating endpoints are only the edge-routers; it is not possible to see any detail.

- Secure Socket Layer (SSL)

The standard and most used communication protocol to realize a VPN is SSL.
SSL 3.0 is an acronym for Secure Socket Layer, has become TLS standard with small changes. Applies security in the Transport Layer.
It was designed originally to provide protection on Web communication, HTTP on top of a protected encrypted transported layer.
TLS is a protocol that place itself in the middle-layer between the Transport Layer and the Application Layer.
When implemented on boundary routers can provide a tunnel between two sites (typically LANs).
It is placed on top of the TCP protocol, there is no need to change anything on the existing layer.

It provides a secure channel (byte stream) between any TCP-based protocol, usually https://, and also other many different protocols NNTP, SIP, SMTP.
There's the possibility of optional authentication.

HTTPS: HTTP on top of TLS

HTTPS is not really a different protocol, it is HTTP but not used as it is on a simple TCP connection, but on a TCP connection in a middle layer of TLS.

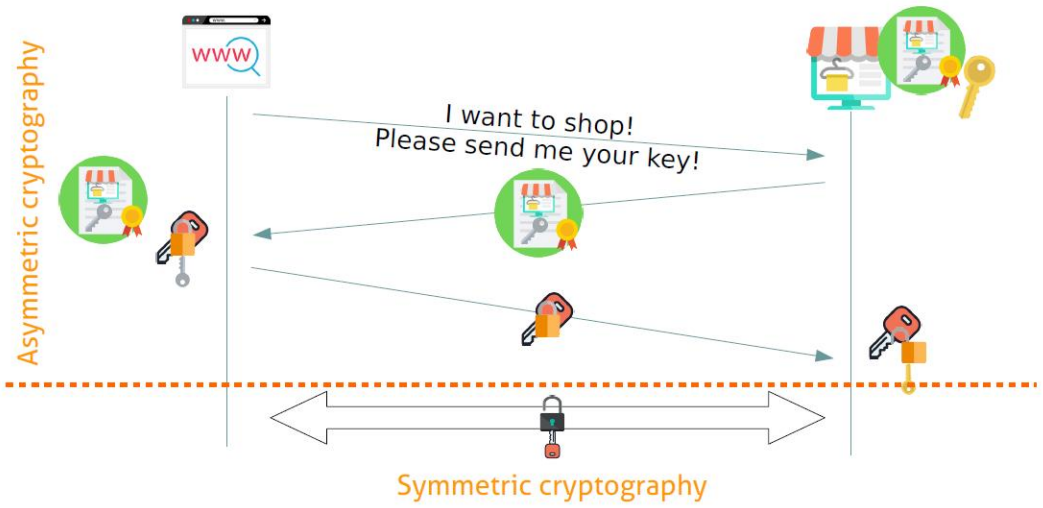


Figure 81: HTTPS, how it works

The host asks for a key/prove of authenticity of the website.
The server sends a *digital certificate*, that is a statement of identity attached with a *public key*, authenticated with a *digital signature of a CA (Certification Authority)*.
The CA receives a request to sign a certificate and issues the digital signature.
The client receives the digital certificate and verifies the authenticity of it, that it is not expired or revoked, and also verifies the digital signature.
If all is okay, then it extracts the public key of the web site.
It uses the public key of the website to encrypt a *session key*, that is sent to the website.
Once received, the website uses its private key to extract the session key.
The session key or the symmetric key, is then used to exchange the real data; all the steps before are used to setup this configuration.
Only after the ***symmetric cryptography*** channel is established, is possible to use the HTTP protocol to exchange data.
The exchange of data never happens using asymmetric cryptography but only using symmetric cryptography that means the same session key; the asymmetric cryptography is used only for exchanging the session key between server and host.
In this not possible to have MITM attack in this schema.
The Certification Authority is the trusted-entity that uses its digital signature to sign the website certificates and in which all devices refers to and trust for this kind of certifications.

SSL protocol Architecture

SSL is a middle layer between Application Layer and Transport Layer.
It is like a general purpose fragment that can realize different tasks.
The standard one is to exchange data, but it is possible also to have SSL Handshake and the SSL Change Cipherspec.
The SSL handshake is used to establish the secure connection, to authenticate the server and to agree on the encryption keys and algorithms that must be used.
The SSL change cipher spec is used to select agreed keys and encryption algorithm.
The SSL Alert is used to exchanging info about failures.

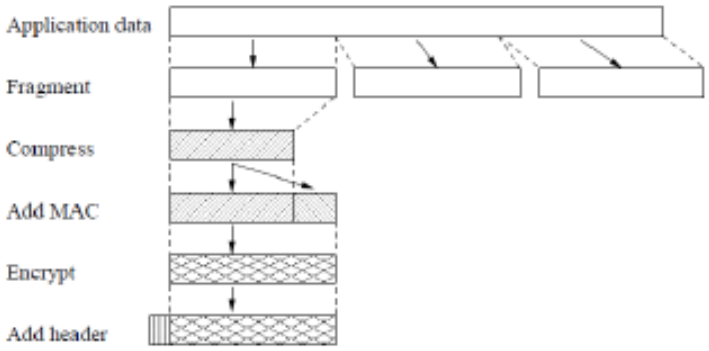


Figure 82: SLL Record Protocol

This is the usual structure of a TLS packet that is sent into the transport layer.
First there is the fragmentation of the application data (handshake, or cipherspec, or real data), then it is possible to compress the fragments.
Then there is the addition of a keyed *MAC (Message Authentication Code)* to the fragment and the encryption using the shared encryption key.
The added header is telling the receiver how to handle the encrypted data.

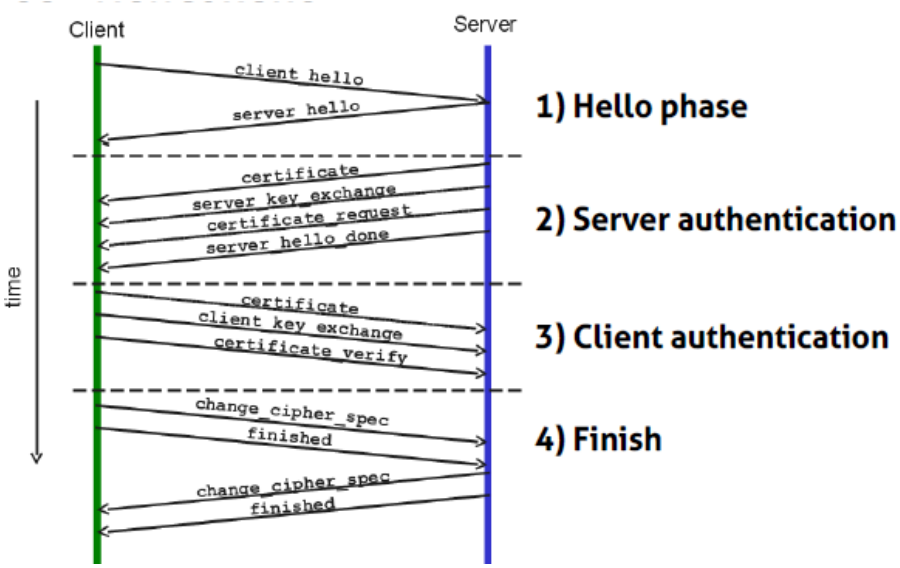


Figure 83: SSL Handshake

In the **SSL handshake** there are 4 initial phases: hello phase, server authentication, client authentication and the final phase.
1) *Hello*: Establishment of security capabilities: Client sends list of possibilities, in order of preference. Server selects one, and informs Client of its choice. Parties also exchange random noise for use in key generation.
2) *Server authentication and key exchange*:
Server executes selected key exchange protocol (if needed).
Server sends authentication info (e.g. **X.509 certificate** to prove its identity), some information used to generate the future session key (*server_key_exchange*) to Client.
Optionally it is possible to have a *certificate request*, that is used to prove the identity of the client, so it used for mutual authentication and not only authentication of the server.
3) *Client authentication and key exchange*:
Client executes selected key exchange protocol (mandatory).
Client sends authentication info to the Server (optional) and the authenticity of the certificate.
4) *Finish:* Shared secret key is derived from pre-secrets exch. in 2, 3.
Change Cipher Spec. protocol is activated.
Summaries of progress of Handshake Protocol are exchanged and checked by both parties.

The **SSL/TLS Security Capabilities** is conventionally expressed by a descriptive string, in which is specified version of SSL, key exchange algorithm, grade of encryption, encryption algorithm, mode of block encryption, cryptographic checksum algorithm. An example of this string is:
TLS_RSA_WITH_AES_128_CBC_SHA

Possible ways of agreeing on secrets in TLS are:
- RSA: RSA key exch. (secret encrypted with recipient's publ. key)
- DHE RSA: Ephemeral Diffie-Hellman with RSA signatures
- DHE DSS: Ephemeral Diffie-Hellman with DSS signatures
- DH DSS: Diffie-Hellman with DSS certificates
- DH RSA: Diffie-Hellman with RSA certificates
- DH anon: Anonymous Diffie-Hellman (no authentication)
- NULL No key exch.

TLS is the successor protocol to SSL. It was introduced in 1999 as an improved version of SSL 3.0 and was initially called SSL 3.1. The current version is TLS 1.3 (as of 2018).
The master secret, that is the way in which the session key is generated by combining different pre-shared information between client and server, is derived in different ways from TLS and SSL.

SSL/TLS Heartbeat, Heartbleed bug

When a connection is established, it is possible to use a feature called heartbeat, that allows to avoid that connection is not used for some time.
For example if the connection is not used for 10 minutes, than also if there is no exchange of real data, a small packet (an heartbeat) is sent only to keep the established session alive.
This in order to avoid the re-negotiation of the security parameters for a new secure session.
This extension introduces the HeartbeatRequest and the HeartbeatResponse.
When one endpoint sends a HeartbeatRequest message to the other endpoints, the former also starts what is known as the retransmit timer.
During the time interval of the retransmit timer, the sending endpoint will not send another HeartbeatRequest message.
An SSL/TLS session is considered to have terminated in the absence of a HeartbeatResponse packet within a time interval

```
struct {
      HeartbeatMessageType type;
      uint16 payload_length;
      opaque payload[HeartbeatMessage.payload_length];
      opaque padding[padding_length];
} HeartbeatMessage;
```

Figure 84: The Hearbeat message structure

In the next figure it is showed how the Heartbleed bug was used to possible capture sensitive information related to session key and passwords; the receiver does not check if the string that must be sent is equal to declared length, and it could send also the memory locations placed next to the string that could potentially contain sensitive data.

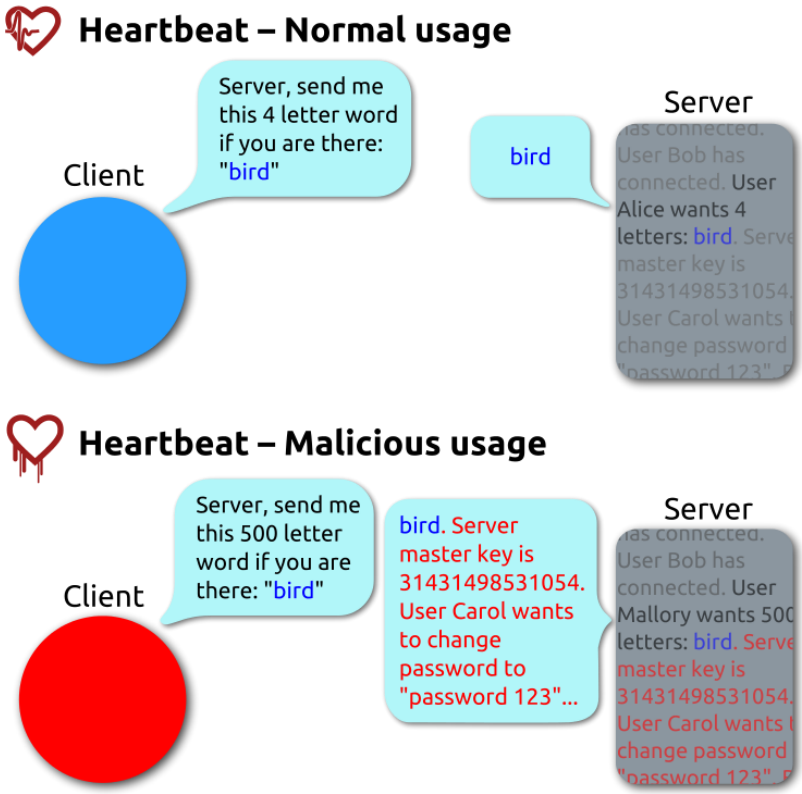


Figure 85: Heartbleed vulnerability

- <u>SLL VPN Architectures, Functionalities, Services</u>

SSL Portal VPN

Allow remote users to connect to VPN gateway using a web browser. Access services from Web site provided on gateway.

SSL Tunnel VPN

Allow remote users to access network protected by VPN gateway.
More capabilities than portal VPNs, as easier to provide more services.

Functionalities:

- Proxying: intermediate device appears as true server to client, like web proxy.
- Application translation: convert information from one protocol to another.
- Network extension: extend the network that one user sees behind the VPN, using typically a Tunnel VPN.

Services:

● Authentication Via strong authentication methods, such as twofactor authent., X.509 certificates, smartcards, security tokens etc. May be integrated in VPN device or external authent. server.
● Encryption and integrity protection: Via the use of the SSL/TLS protocol.
● Access control: May be per-user, per-group or per-resource.
● Endpoint security controls: Validate the security compliance of clients attempting to use the VPN. – e.g. presence of antivirus system, updated patches etc.
● Intrusion prevention: Evaluates decrypted data for malicious attacks, malware etc.

Considerations

Encryption and cryptography is good but is not sufficient for Web Security.
The server, if there is no mutual authentication in SSL and no client-side certificate, does not know anything about the client, except the IP address. SSL provides only a secure channel, but you do not know who is at the other end of the tunnel.
The client instead receives the server's certificate.
A certificate means that someone attests to binding some name to a public key.
Every browser has a list of built-in certificate authorities. It's all a matter of trust...
There is no rational basis for deciding whether or not to trust a given CA.

XII. Proxies

- History

It is a very old idea, it has been developed 30 years ago, in 1994.
The original definition of Ari Luotonen in 1994 is that a *"WWW proxy server provides access to the Web for people on closed subnets who can only access the Internet through a firewall machine".*
The original idea is that *"An application-level proxy makes a firewall safely permeable for users in an organization for users in an organization, without creating a potential security hole through which "bad guys" can get into the organizations' net."*
Namely: one single host handling requests from several users.

- Difference between Proxy and VPN

When we refer to the application-level it means that we are dealing with the *real data*, and we are not dealing with packets that have to been forwarded or to been send somewhere. This is the main difference, also with respect to VPN.
VPN usually is not really intended to read content of the packets; with proxy it is a real-data exchanging.
The proxy must do a request in place of a host, this is the main difference.
A proxy has to deal with the request of the clients and has to understand the requests. It is responsible for the application that has to receive data, proxy usually have to deal with semantic of request and understand what a client wants.
VPN usually works at lower layers and do not really read the content of the request.

- Proxied HTTP transaction

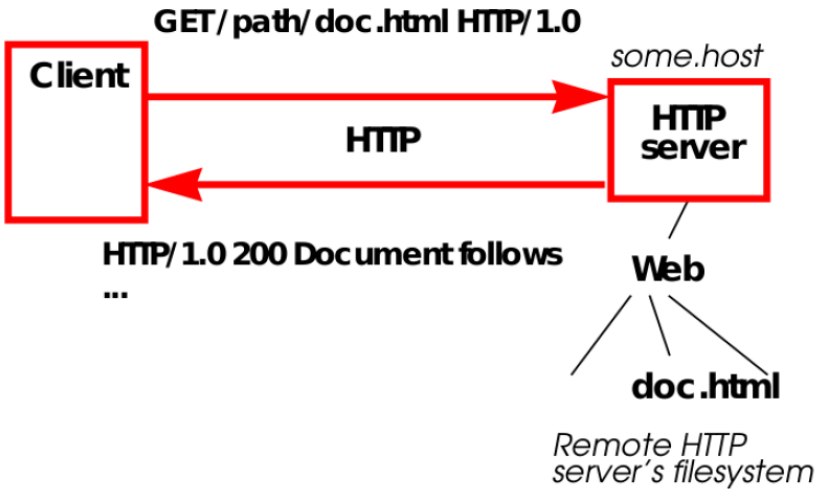


Figure 86: Normal HTTP transaction

In a normal HTTP transaction, there is a client that has to connect to an host that hosts an HTTP server.
What happens is that there is an HTTP exchange with a REQUEST from the client (GET), then the server receives the packets related to the request and reads the application-layer data, then it goes into the filesystem, reads the data requested that must been sent, and sends data to the host in a series of packet (HTTP/1.0 200).

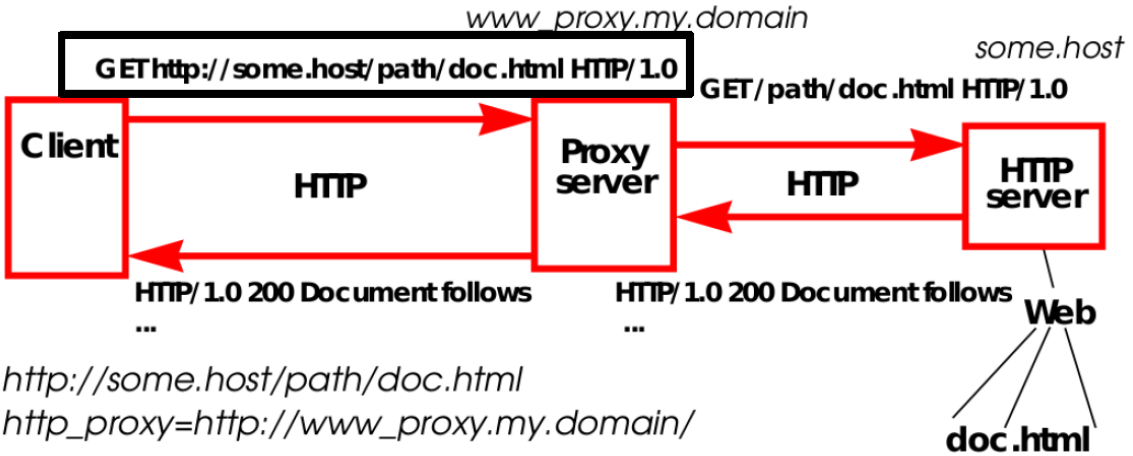


Figure 87: Proxied HTTP Transaction

In a proxied HTTP transaction, the real request is not done by the client itself but by an intermediate host, the Proxy Server.
It receives the request from the client, reads the request and performs the real request to the HTTP server.

There is no effective interaction between client and HTTP server; the Proxy server mediates the transaction and realizes the exchange.
The host sends the HTTP request to the Proxy, the Proxy reads the HTTP request (application-layer payload), and then performs the real request.

The main difference with the HTTP standard exchange is that in the HTTP Request for a Proxy Server is a bit different because it includes the host into the request, so there is no use of relative path like before but the **absolute path**. In this way the proxy can extract from the HTTP request the destination host to which the request must be forwarded.

The proxy has to be able to handle different requests, for performing other kind of protocols requests like FTP request and FTP response.
It is responsible for the application that has to receive data, proxy usually have to deal with semantic of request and understand what a client want.

- Benefits of Forward Proxy

- ***One single rule on the firewall***: only the proxy can access any host outside the network (Internet), all other hosts in the network are not allowed to reach Internet.
- ***Authentication***: with a proxy you have point when you can identify the identify of users; for example, if proxy gets a request, it can ask for the authentication of the user that requests.
- ***Authorization***: it is possible to see if the users is able to do some operations like access to the Internet or it is not authorized to do this.
- ***Auditing***: it is possible to log every action of users, like which resources they have accessed and at which time.
- ***Whitelisting***: allow only some resources or websites that could be accessed outside, every else is denied.
- ***Blacklisting***: the other way around, here is listed all resources that are denied.
- ***Caching*****:** this was crucial in the past (1994) when the only connection was at kbit/s. So whenever you must download a single image, it takes 30 seconds.
The idea was to reduce the bandwidth with the server by caching and storing all the documents and files in the proxy, that is placed in the LAN that is much faster than going outside, and there was no need to pay connection because it's internal.
The problems that arise with caching is how to know if the document in the cache is up-to-date; the solution was the *If-Modified-Since* request header that simply is the date of the last update and then it is possible to compare it with the date of the cache document.

- ## Forward Proxy

The **HTTP request** for a Forward Proxy is a standard request in absolute-form to the proxy. Then the proxy, extracts the name of the host, and forwards the request towards the final destination. The proxy is the middle-point.
The forward proxy is placed in the internal LAN, and not anywhere in the Internet.

Moreover, it can be used to have an **HTTP tunneling** by the use of *HTTP CONNECT*.
HTTP CONNECT is a special case of request from HTTP, so that it is possible to forward any kind of connection using an HTTP request.
With HTTP CONNECT it is possible to use any protocol that uses TCP.
The proxy establishes the TCP connection, so now the focus is not on application layer but on lower layer, the transport layer.
By establishing a TCP connection, the proxy doesn't go anymore into the details of the exchange between client and server, because it only forwards stream of bytes.
The proxy receives simply the destination host that the client wants to connect to and establishes the connection on its behalf; it will not forward an HTTP request.
When the connection is established, the proxy server is a middle point that continues to forward the **TCP stream unmodified** to and from the client.
This is really a way to give access to any possible service outside the network.
The HTTP CONNECT is very powerful, for example if you want to bound the internal hosts to use any different protocol than HTTP/HTTPS, with this HTTP CONNECT it is possible to bypass the firewall because in this way can use any protocol.
If for example you want to deny SSH connection, but you allow HTTP CONNECT, it could happen that the host connects to the proxy using HTTPS protocol, and inside the connection there's the use of SSH.
That is why it is called *HTTP tunneling*, HTTP is the tunnel used for another kind of connection, like SSH or any other type of TCP-based protocol.

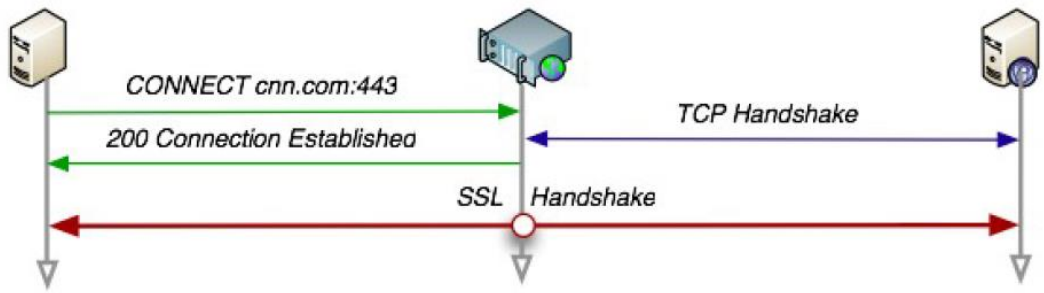


Figure 88: HTTP CONNECT mechanism

Looking at the picture, imagine that you want to allow using the HTTP CONNECT to SSH from the client to the final server.
But you want to use the proxy to filter, audit, authorize and authenticate.
The client issues HTTP CONNECT request to cnn.com on port 443; the proxy receives the HTTP CONNECT request and establishes the TCP Handshake and replies to the client with 200 Connection Established.
Finally, there is the real data exchange using the decided protocol.
The proxy is not anymore supposed to understand the data that client and server are exchanged; it blindly forwards what it receives from the client to the server and the other way around.

The HTTP CONNECT must be used with attention, and this is why not all proxy servers support it or at least to limit it to port 443 only.
Because if you use a CONNECT to any port, then you potentially allow the client to use any TCP-based protocol also if maybe it is blocked by the firewall.

- Content-filtering proxy

It is strictly related to authentication and authorization: after the user authentication, the proxy can control what actually it is sent or received.
It is used to filter the content like no Facebook or no porn websites and this could be based on blacklists or semantic searches.
Other kind of filtering are related to virus or malware scan, or to check that for example there is no transmission of executable/binary files or watermarking.

- Anonymizer proxy

It is a special type of proxy, that is not really supposed to be placed and used in the internal network, but it is something like a proxy server anywhere in the web that acts as an intermediary and privacy shield between the host and rest of the Internet.
It changes the original source of a request to a server; the real server will not know that the host requests that page, the anonymizer proxy breaks the link between client and server.
The idea is that the final connection with the server, is not been done by the client but by the anonymizer proxy.
Typically this is used for accessing restricted content, like thepiratebay, or to bypass filtering based on IP glocalization.

- SSL Forward proxy

Using an SSL connection between a client and a server, a proxy in the middle is not able to understand what data are exchanged, because they are encrypted using a symmetric key that is exchanged between client-server in the 1st phase.
So, if the network administrator wants to know and to filter the content that hosts and server are exchanging, it must use an **SSL Forward proxy**.
It is a way to decrypt and inspect SSL/TLS traffic from internal users to the web, generally implemented in firewalls.
The SSL Forward Proxy decryption prevents malware concealed as SSL encrypted traffic from being introduced in the network; this is used to perform IDS.

– The proxy uses certificates to establish itself as a trusted third party to the session between the client and the server.
– As the proxy continues to receive SSL traffic from the server that is destined for the client, it decrypts the SSL traffic into clear text traffic and applies decryption and security policies to the traffic. If it is authorized or not.
– The proxy, then, re-encrypts and forwards the traffic to the client

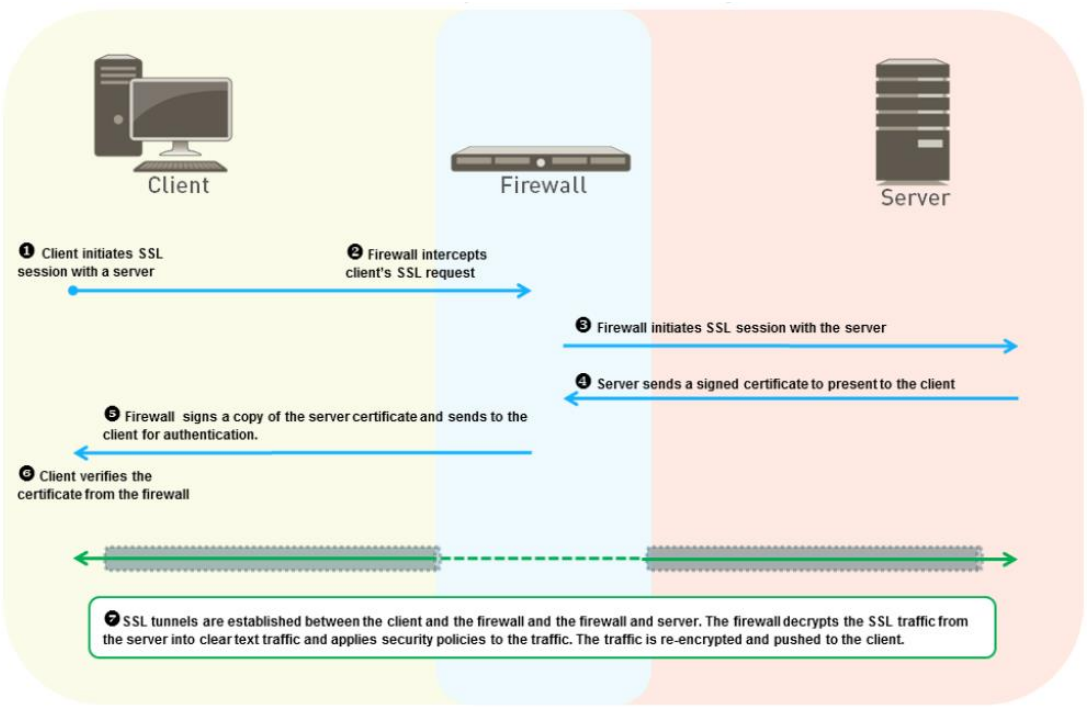


Figure 89: SSL Forward proxy

- Reverse proxy

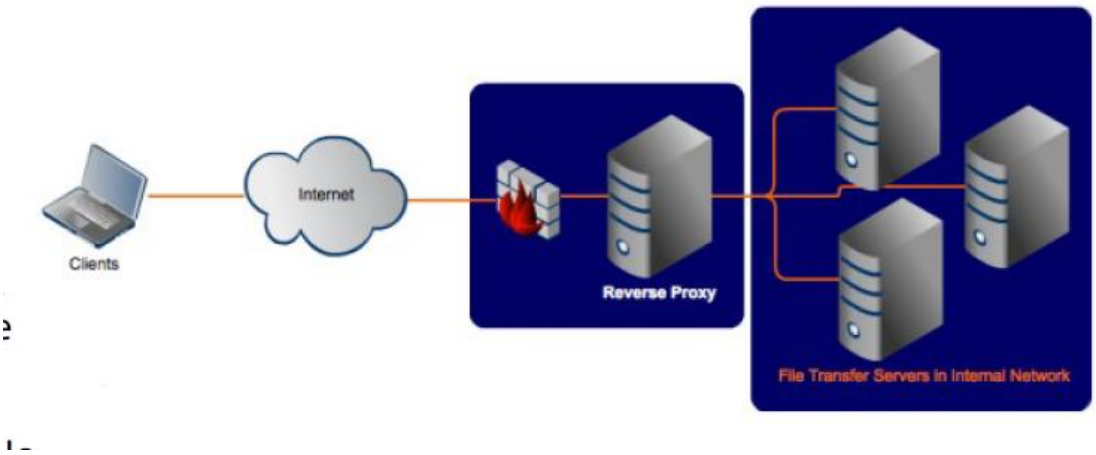


Figure 90: Reverse proxy

Reverse proxies are similar in the concept of forward proxies.
But while forward proxy operates on behalf of the clients, reverse proxy operates on behalf of the server.
The client connects to the proxy, and it thinks that is connected to the server.
From the point of view of the client everything is transparent; it thinks that is connected to the server but at the very end is connected to the proxy.

The **reverse proxy** receives the requests from the outside as if it were the server and then forwards the request to the actual destination server.
It is supposed to understand the request, so at application-level, and then according to it, forwards to the real destination.
Typical functions of reverse proxy:
- *Load balancing*: instead of forwarding all the requests to one single server, they are divided and forwarded to different server.
- *Cache static content*: the proxy stores locally all the static content (pictures, icons) and the dynamic content is taken from other server like database servers.
- *Compression*: the overhead is split between server and proxy
- *Accessing several servers into the same URL space*:
- *Securing of the internal servers*
- *Application level controls*
- *TLS acceleration:* it is possible to perform the encryption at proxy level, without giving this task to the servers and have less overhead.

- Internal server protection

The reverse proxy receives the requests from the clients; then it can *clean the request* and then issues new and proper requests to the real server.
For example, if there are queries to a database, it is possible to have an SQL Injection by using not appropriate characters.
The reverse proxy could receive this request, performs the sanitization by filtering the input and removing dangerous content for the internal server, and forwards the clean request; or it is also possible to receive a request and forwards to the server a totally different request.
Also, there is no direct connection between internal hosts and Internet and this is a defense against Denial of Service attack.

It can also provide support for HTTPS to servers that only have HTTP.
It is possible to add AAA (Authorization, Authentication, Auditing) to services that do not have them, for example a IoT device behind a firewall that must be accessible from the Internet.

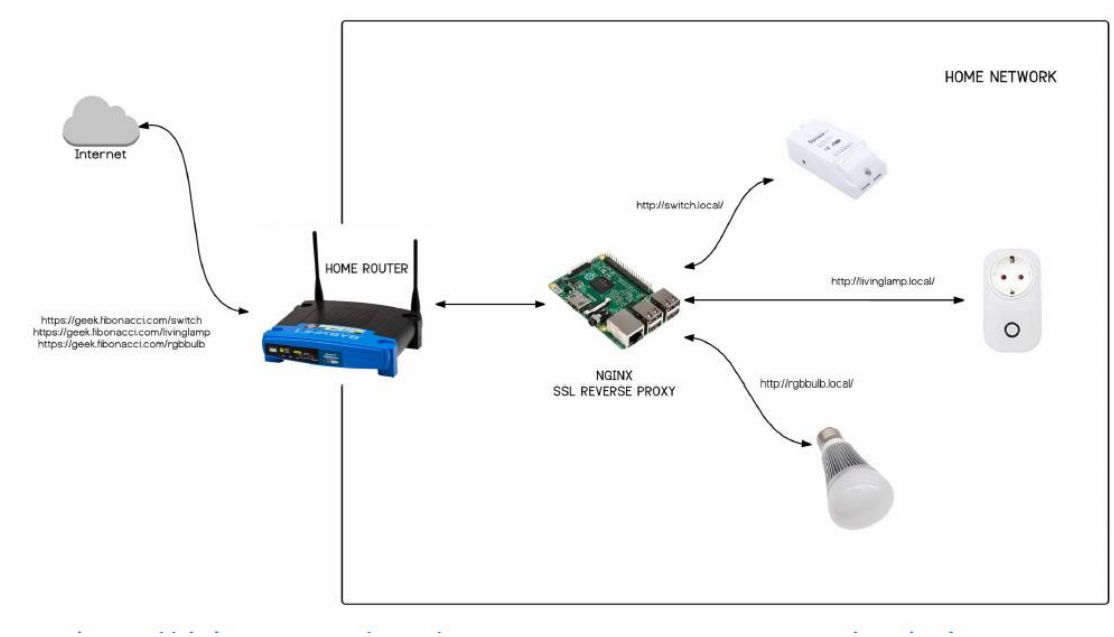


Figure 91: Reverse proxy performs AAA for IoT devices

- ## Reverse proxy for application control

Application layer firewall operates at the application layer of a protocol stack.
A **WAF (Web Application Firewall),** for example a special module of Apache Webserver ModSecurity, instead of reading the request as it is, it inspects the HTTP traffic by applying some policies of sanification.
It prevents attacks such SQL Injection, Cross Site Scripting (XSS), file inclusions, and other types of security issues.
It can block application input or output that is malformed, like strange characters, or values not supposed to be used, or block contents that violate policies.
Also, another powerful tool is that it is possible to enforce detection whether an unwanted protocol is being provided through on a non-standard port or whether a protocol is being abused in any harmful way.

- ## Reverse proxy for TLS acceleration

The SSL/TLS “handshake” process uses digital certificates based on asymmetric or public key encryption technology.
Public key encryption is very secure, but also **very processor-sensitive and thus has a significant negative impact on performance causing SSL bottlenecks**.
Then the idea is that to enforce a reverse proxy that performs TLS acceleration: it has a special hardware support to perform modular operations with large operands.

SSL offloading
There is an *SSL Termination* that is the reverse proxy. It decrypts the TLS/SSL-encrypted data and the sends it in clear text to the server.
This is good because it also allow the possibility of an IDS or application firewall inspection that checks the type of data sent into the network.

SSL Forwarding (or Bridging/Initiation)
The reverse proxy intercepts and decrypts the TLS/SSL-encrypted traffic, examines the contents to ensure that it does not contain malicious code, then re-encrypts data before sending it on to the server.
In this case, the traffic between proxy and server is encrypted again and no in clear text.

- ## Proxy and HTTPS

HTTPS traffic cannot be read.
If we want to read the client HTTPS traffic, we need to perform a MITM attack, pretend to be the real server and be the termination of the SSL/TLS connection.

This is implemented in proxy using a technique called **SSL bump**.
The idea is that the proxy dynamically generates real-time the server certificate, with the proxy public key, so that can impersonate the named server.
In this way the client establishes a connection with the proxy, but it will not complain because the certificate is valid also if the public key is the one of the proxy server.

To generate dynamically the certificate, *the proxy needs the host-name of the final server*.
But this could be a problem because the TLS handshake is started using the IP address and the hostname is sent after, in the HTTP request.
For example, imagine that there is a server, like amazon.com, that usually can host multiple websites: the same IP address could host potentially hundreds of websites.
Before the first GET request is sent, it is needed to establish a TLS connection.
The hostname field, that is the specific website to which the client is trying to connect using HTTPS, is in the HTTP request *that comes after the TLS connection is established.*
But the TLS connection, that comes before the GET request, requires that the server sends a certificate to the client related to the specific website it has to visit.
But at this point, how the server can know which certificate has to be sent to the client (because it could host hundreds of websites) ?
This cannot be solved without having the name in advance, so before starting.
There is the need an extension of TLS, called **Server Name Indication (SNI)**, in which the client indicates which hostname it is attempting to connect at the start of the TLS handshaking process, and it is in clear text.
This allows a server to present multiple certificates on the same IP address and TCP port number without requiring that all those sites to use the same certificate.

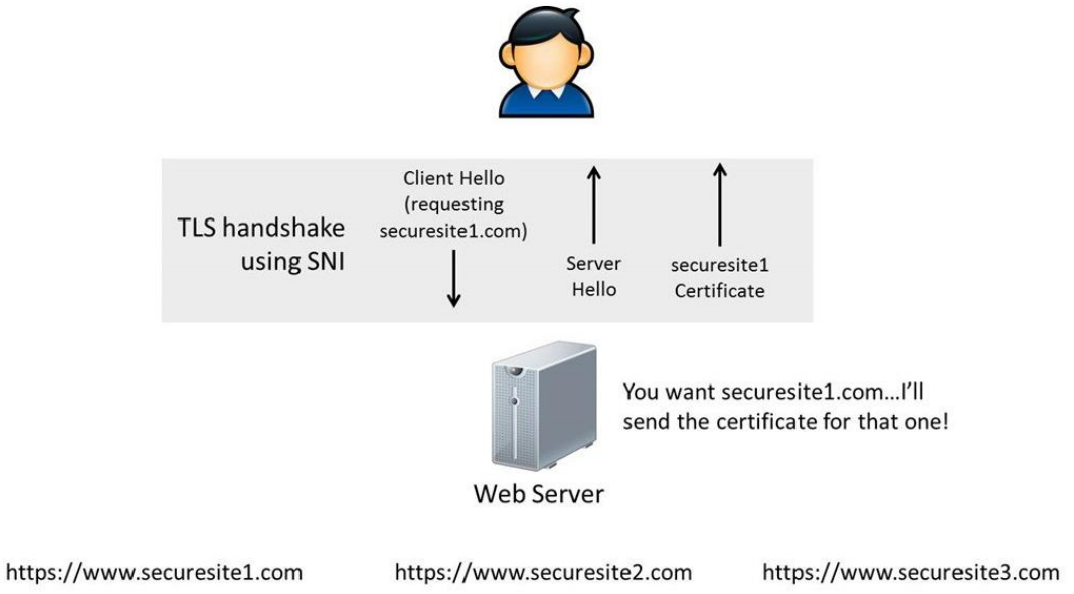


Figure 92: SNI before the TLS handshake

The Web Server in the figure hosts 3 different websites; the client wants to visit the securesite1.com. So the webserver needs this information before the TLS started, because it has to forward to the client the right certificate.

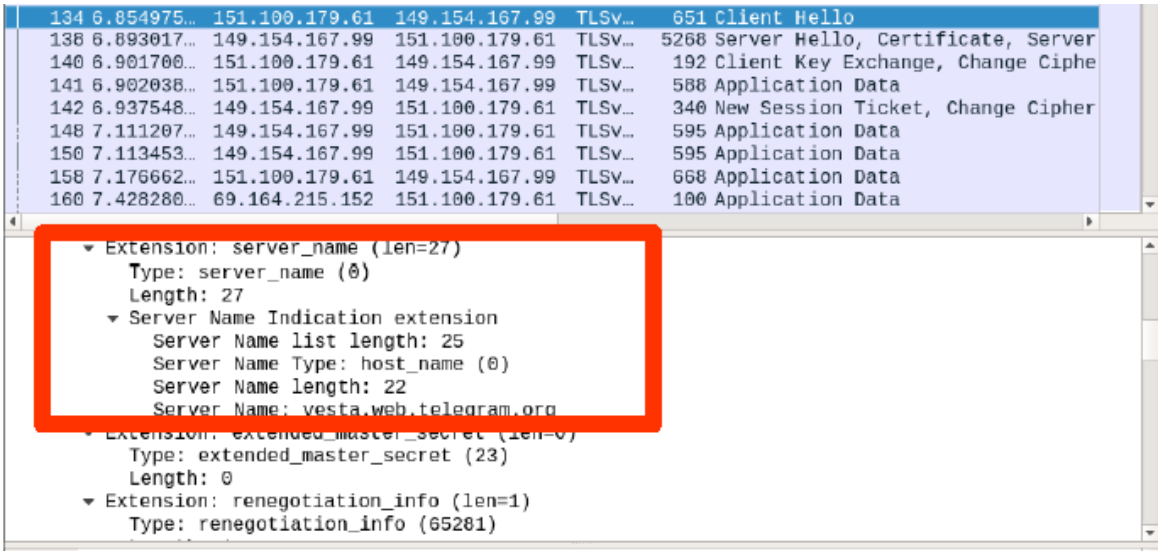


Figure 93: The SNI indication in the Client Hello message

The possible consequences of the SNI is that it is in plain text, so it is possible for an eavesdropper can see which site is being requested.
This helps security companies provide a filtering feature and governments to implement censorship.
There is a trick called *Domain Fronting trick*, in which in the SNI there is the use of

a generic name for which it is used a certificate that will be valid inside that domain, it is like having a white-card.
There is also an upgrade called ***Encrypted SNI (ESNI)*** in which the hostname is sent to the server in an encrypted way; so it will be clear only when the TLS connection is established from the client.

- SOCKS proxy

It is a circuit-level gateway and works between application layer and transport layer.
It is very similar to the HTTP CONNECT proxy mechanism, but it is more versatile.
It is not really stick-on HTTP. The difference is that HTTP CONNECT works at higher layer (application-layer), SOCKS proxy works in the intermediate layer between app and transport layers.
It can enforce different authentication mechanisms, it can also tunnel TCP but also UDP and IPv6 (SOCKS5), and can also work as a reverse proxy.
SOCKS proxy is implemented in SSH, putty and Tor.
You can relay any protocol wanted using a SOCKS proxy.

- Transparent proxy

One of the characteristics of all proxies that we mentioned before is that the client should be aware of the presence of proxy.
The idea of transparent proxy is to provide the same features but without clients being aware the presence of it. It should be something like to intercept the request and try to relay it even if it was not structured as a proxy request.
The proxy is transparent because intercepts the requests, and provides an answer if is needed that is completely transparent to the client.
A transparent application proxy is described as a system that appears like a packet filter to clients, and like a classical proxy to the server.
Alternative names for transparent proxy are intercepting proxy, inline proxy or forced proxy. It is a common practice for many ISPs and mainly for mobile users.
For example, TIM Mobile used this transparent proxy to send compressed versions of images that were accessed by the mobile customers; because the quality of the display is not so high to receive the entire quality of the image. It also performs some caching of data because it is very probable that many customers of TIM request the same resource.

At the start of session, a TCP packet with a source address of client and a destination address of server reach the proxy system, expecting to cross it just like a standard gateway.

The proxy's TCP/IP software stack check if the incoming packets has a destination address that is not one of its own addresses.
The proxy will accept and process the request as it is directed to itself; so it pretends to be the server, reads the content and evaluates the request. Only a this point it decides to forward or not, by modifying obviously all needed fields like the source.

The difference with respect to NAT is that the NAT only switch the IP addresses; in the case of the transparent proxy is really a different request, so we are creating a different packet.

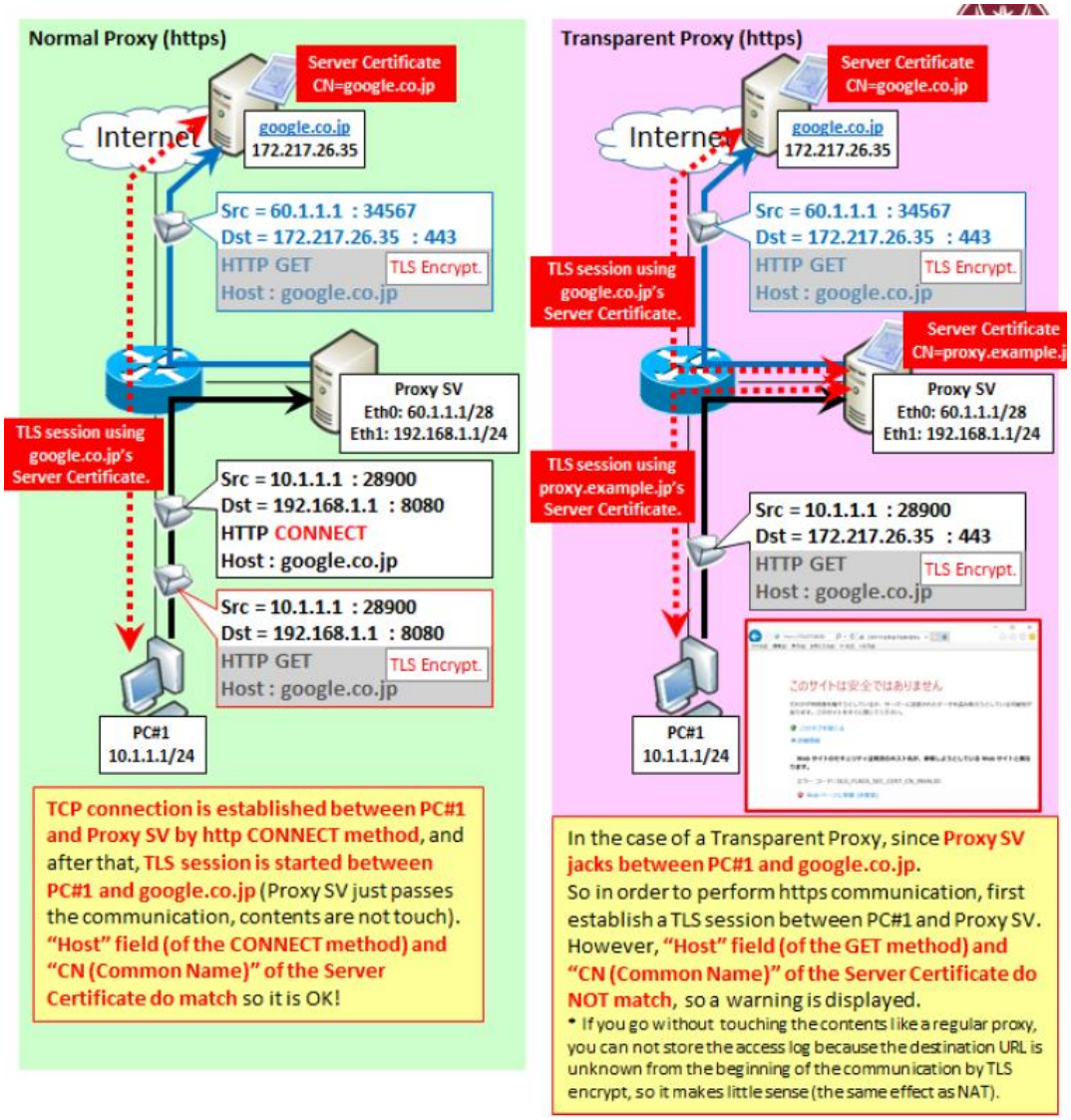


Figure 94: Difference between normal and transparent proxy

When an host run a normal proxy, the client has to configure the parameters to use the proxy; when an host use a transparent proxy there is no setup in the client.

In the standard proxy mechanism, the source of the request is the client and the final destination is the proxy itself; inside the request there is the final destination. Then the proxy receives the request and generate a new request with its own IP address and destination the final one.
In the transparent proxy mechanism, the source of the request is the client and the final destination is not the proxy this time but is the server; the transparent proxy intercepts the packet and generate a new request with its own IP address.
The DNS request is performed by the proxy in the standard proxy mechanism and by the client in the transparent proxy case.

Limits of Transparent Proxy
The transparent proxy intercepts the HTTP request that has the IP address of the server. Using HTTPS there is a certificate exception because the certificate that the client receives is not the real one.

To be able to realize a real transparent proxy we should forward a subset of packets to transparent proxy and others must be forwarded as they are without do nothing. If there is interception at the default route, we will break some protocols like SMTP, POP and IMAP.
So the solution is to use **Policy-based Routing (PBR).**

Policy based Routing PBR
The definition of routing is to take decision among different possible directions on which path to forward a packet.
Traditionally routing is a *"destination-driven process*": when the packet is received, you apply the network mask, and apply the longest prefix match rule to decide where to forward the packet.
But it is possible to take routing decisions with respect to other features and options: this is the idea behind the PBR.
It is possible to use as factor not only destination, but also by considering other elements to decide the next-hop of a packet: quality of service, source routing, traffic shaping.
PBR goes beyond simple destination-drive routing.

Returning to the transparent proxy question, the PBR is applied considering the type of protocol that is used inside the request.
For example, if the protocol is HTTP then it is possible to redirect the packet to the transparent proxy. But if there is some protocol that cannot be accessed by the proxy, then the packet must be dropped or forwarded as it is.

This could be done using iptables and a process that is called *“packet marking”* using the mangle table.

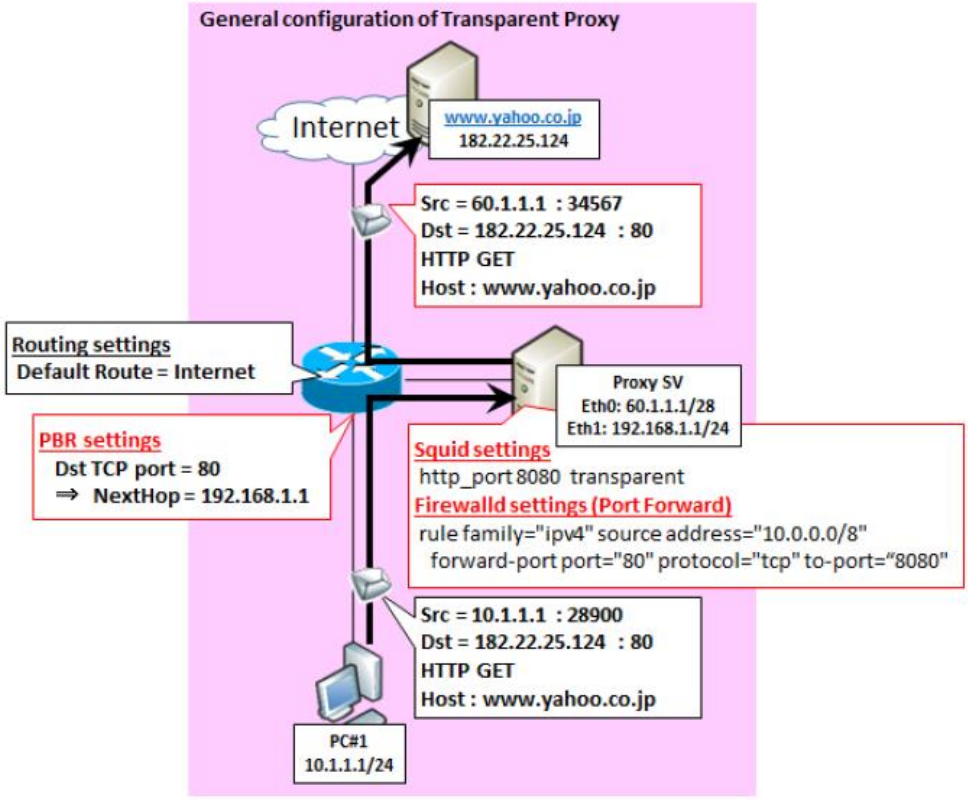


Figure 95: General configuration of Transparent Proxy

When there is a request with port 80, then the transparent proxy intercept it; with other protocols don’t intercept.

- ## ICAP: Internet Content Adaptation Protocol

The proxy is related to the application layer and real data.
The ICAP is based on the idea of a protocol that can be applied in order to make some transformations of the content (data) of the packet.
ICAP is defined in RFC-3507; it is a protocol aimed at providing simple object-base content vectoring for HTTP services.
In essence it is a protocol for executing remote procedure call on HTTP messages.
It allows ICAP client to pass HTTP messages to ICAP servers for some kind of transformation or other processing, called “adaptation”.
ICAP is a way to specify how you can handle some data, executes its transformation service, and sends back responses to the client usually with modified messages; it could be for example a service of compression of the data.

It is possible to see the power of this idea using in combination with a transparent proxy: it allows to transparently provide additional functionalities, using standardized interface towards ICAP servers.
This is used also by ISP, like TIM, as we said before with compression.
Examples of ICAP transformations:

- *Simple transformations of content can be performed near the edge of the network instead of requiring an update copy from the origin server*: translation to other language, formatting for different devices, different encodings, different advertisements;
- *Expensive operations on the content by dedicated software*: file scan for viruses
- *Checking if requested URIs are allowed or not*: parental control, content filtering (content violating digital rights, porn images, and so on).

XIII. IPsec

- IPsec services

IPsec was a network layer protocol suite for providing security over IP.
It was part of IPv6 specification and is an addon for IPv4.
IPsec is made mainly by 3 big protocols that realize different tasks related to security:

- **Internet Key Exchange (IKE)**: this is the most characteristic of IPsec and is just a protocol made for decide and agree on the security parameters of a IPsec connection.
  You can imagine as the first part of TLS, where the two endpoints decide the parameters of the connection, this is very similar.
  There is the exchange of proposals, decisions about details of protection of the channel and then they start using the channel.
- **Encapsulated Security Payload (ESP)**: support for encryption and optionally authentication.
- **Authentication Header (AH)**: support for data integrity and authentication of IP packets.

You can think for a logical model for realizing and using these protocols.
Whenever we have to establish a secure communication between two endpoints there is the need to decide the details of the communication; the details related to a specific channel are called **IPsec Security Associations SA**.
The SA are made of the parameters of encryption algorithms (AES, SHA1, etc…), modes of operations (CBC, HMAC, etc.…), key lengths and type of traffic that should be protected.
The SA has to be agreed by both the two endpoints.
Remember that for a 2-way communication there is the need of two SAs that must be defined: one from host A to host B, and the other one from the host B to the host A.
SA parameters must be negotiated using IKE between sender and receiver before the secure communication starts.
The hosts maintain a database that contains all the SAs needed for all the defined connection between other nodes of the Internet.

Each SA is identified by:

- *Security Parameters Index (SPI)*: 32-bit integer chosen by the sender.
  This allows the receiver system to select the required SA among all the SAs entries in the database of the receiver.
- *Destination Address*: only unicast IP address are allowed.
- *Security Protocol Identifier*: AH or ESP.
- *Other security parameters*: protocol modes (tunnel/transport), protocol algorithms, security association lifetime, replay protection.

• IPsec modes

Transport Mode

Provides protection of a Transport Layer packet embedded as payload in an IP packet. The protected payload is something that belongs to the host itself; it is like an host that wants to send its own packet in a protected way.

Tunnel Mode

Provides protection for an IP packet embedded as payload in an IP packet. In this mode, we act as gateways between networks.

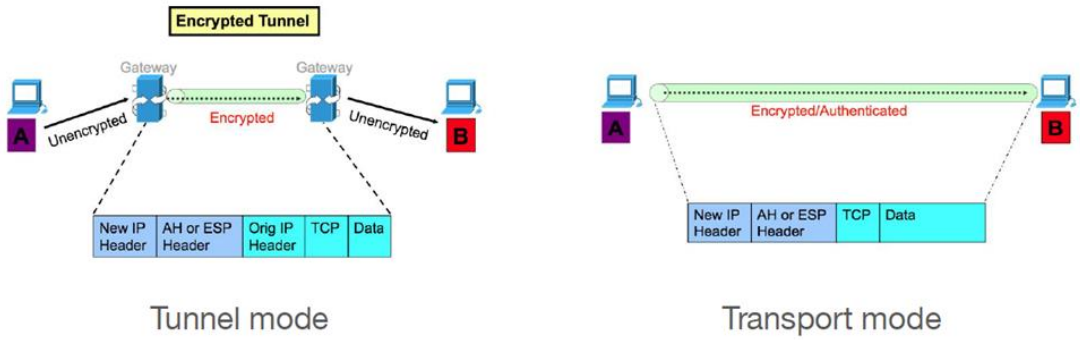


Figure 96: Tunnel and Transport Modes

Transport mode is used generally for end-to-end communication (client-server or two hosts).
Tunnel mode is used when one or both ends of an SA are a security gateway; this way a number of hosts on networks behind firewalls may engage in secure communications without implementing IPSec by itself.
All is realized at network layer.

	Transport Mode SA	Tunnel Mode SA
AH	Authenticates IP payload and selected portions of IP header and IPv6 extension headers	Authenticates entire inner IP packet (inner header plus IP payload) plus selected portions of outer IP header and outer IPv6 extension headers
ESP	Encrypts IP payload and any IPv6 extension headers following the ESP header	Encrypts entire inner IP packet
ESP with Auth.	Encrypts IP payload and any IPv6 extension headers following the ESP header. Authenticates IP payload but not IP header	Encrypts entire inner IP packet. Authenticates inner IP packet

Figure 97: Characteristics of AH and ESP in the different modes

- ## Authentication with IPv4

Depending on the mode used, transport or tunnel, the AH header is inserted after the outermost IP header.

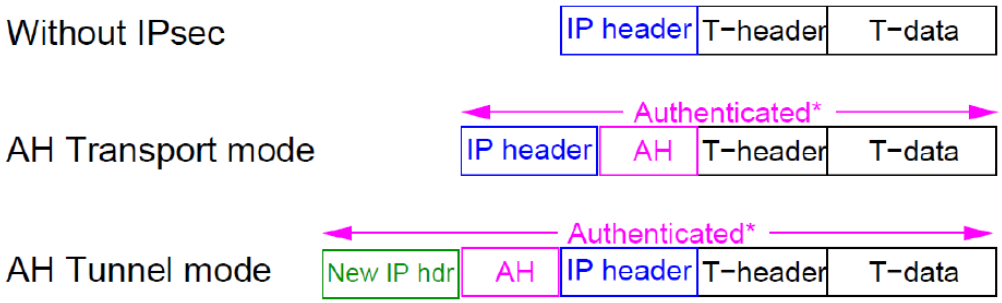


Figure 98: Authentication in IPv4

There are fields that change in an unpredictable way by nature like Time To Live or the Header Checksum and cannot be authenticated. There are fields that instead change but are predictable like the Destination address.
In Transport Mode, the AH Header is placed after the IP Header.
In Tunnel Mode, the original packet is placed as it is as a payload for a new IP header with the AH Header. In this case all the payload is authenticated by the AH Header.

- ## Authentication with IPv6

AH header is inserted after the outermost IP header, depending on the mode Transport and Tunnel mode is used.

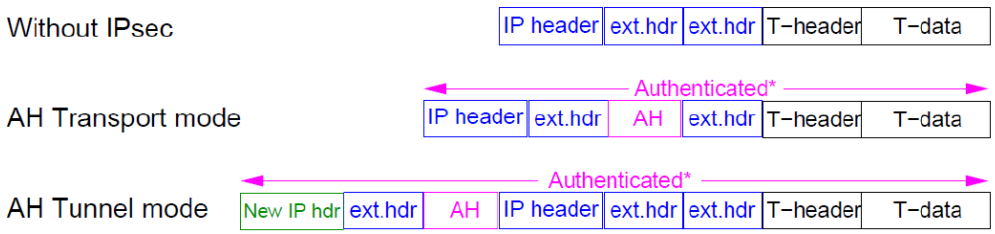


Figure 99: Authentication in IPv6

IPv6 was designed with the extension headers mechanism; some of the extensions header are related to the IPSec AH.
In Transport Mode, the AH IPSec header is directly inserted between the original IP header and the original payload, so as one of the extension header as the natural implementation by design of the extensions mechanism.
AH in transport mode authenticates the IP payload and selected portions of IP headers and IPv6 extension header.
In Tunnel Mode, the original packet will be used as the payload of a new IP packet (it is called inner packet).
The security header AH is going to be inserted between the header and the payload, included in the outer header. Original packet is called inner packet.
The use of AH with tunnel mode provides integrity and authentication of the full original packet (header plus payload) plus some fields in the header of the outer packet.

- ESP with IPv4

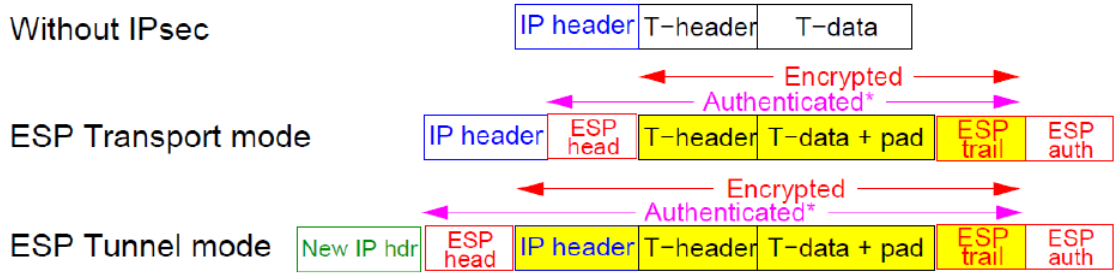


Figure 100: ESP with IPv4

ESP Header is used to encrypt and protect the content.
ESP Header is inserted after the outermost IP header depending on what mode is used.
Padding is added to end of T-layer payload to give (a certain amount) of traffic analysis protection.
ESP trailer and (optional) ESP authentication field added after the end of the padded T-layer payload.
As usual, integrity check (and thus authentication) does not cover any mutable, unpredictable header fields.

It is possible to include some padding, because using symmetric cryptography and using block cipher, it will encrypt block of a given size, so this padding is needed to reach the correct size.
In Tunnel mode the original IP header is encrypted.

- ESP with IPv6

ESP Header is inserted after the outermost IP header, depending on the mode used.

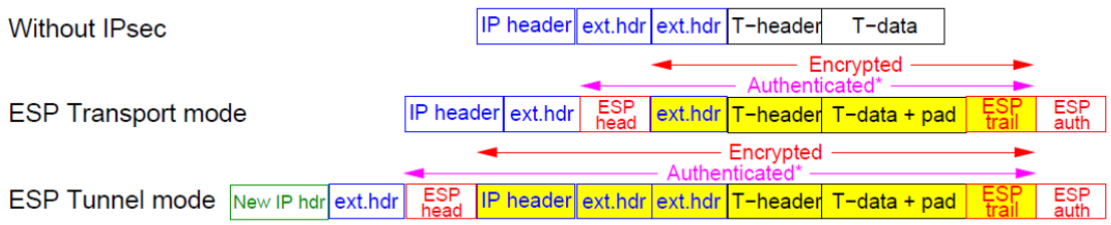


Figure 101: ESP with IPv6

- Encryption plus Authentication

To be finished

- IPsec vs TLS

TLS is much more flexible because is in the upper layers: remember that IPsec must be executed in the kernel space.
TLS provides application end-to-end security, and is the best for web applications for example using the HTTPS protocol.

IPsec is much more complex and complicated to manage with and is very error-prone.

- ## Fundamentals of IPsec

Data origin authentication
– It is not possible to spoof source / destination addresses without the receiver being able to detect this
– It is not possible to replay a recorded IP packet without the receiver being able to detect this

Connectionless Data Integrity
– The receiver is able to detect any modification of IP datagrams in transit

Confidentiality
– It is not possible to eavesdrop on the content of IP datagrams
– Limited traffic flow confidentiality

Security Policies
– All involved nodes can determine the required protection for a packet
– Intermediate nodes and the receiver will drop packets not meeting these requirements

- ## Security Policy

The **Security Policy** is used to define which traffic should be protected using the specific Security Association. It specifies which security services should be provided to IP packets and their details.
With the same SA it is possible to protect different kind of traffic.
The several different Security Policies are stored in a database called **Security Policy Database SPD**.

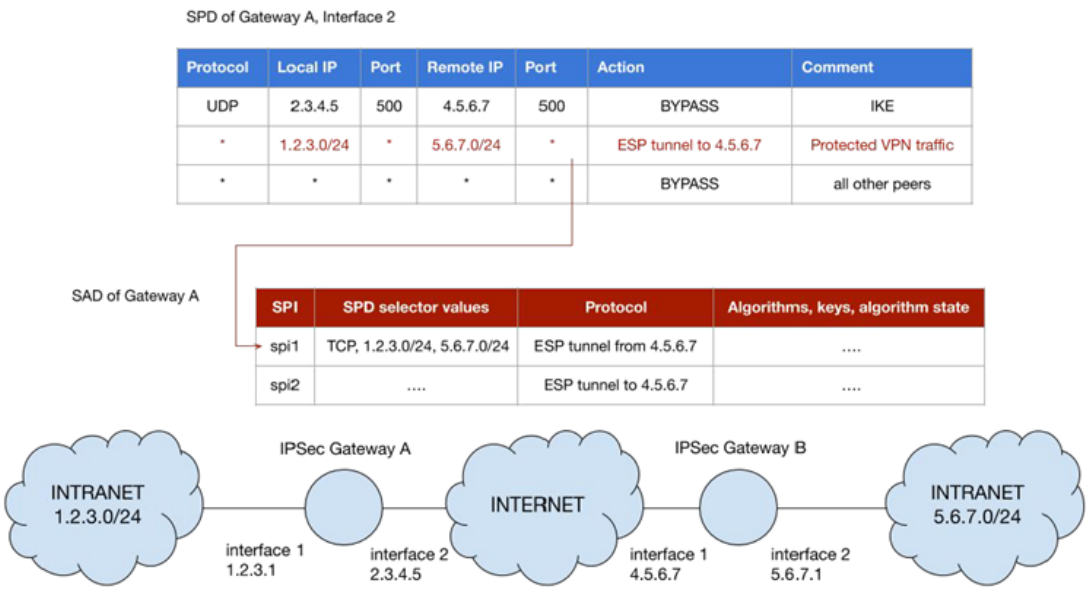


Figure 102: Example of SPD and SAD databases

For each stream, it includes several security attributes: security protocol (AH or ESP), protocol mode (transport or tunnel), other parameters like policy lifetime, port number, and actions (discard, bypass, secure).
A SP is related to a specific IP stream.

- IPsec architecture

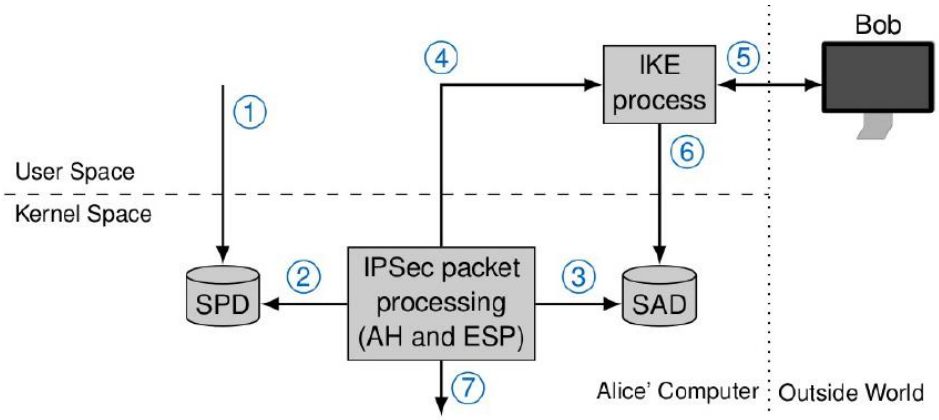


Figure 103: IPsec architecture

Alice has to send a packet to Bob.
In the user space, a packet needs to be sent to the Bob host.
In the SPD database, there's a security policy that states that this kind of packet belongs to a protected traffic using IPsec with encryption (ESP) or authentication (AH).
The IPsec processing module looks to the SPD to decide on applying IPsec protection on the packet. If IPsec is required, so there's an entry in the SPD, the

IPsec secure module looks for an associated IPsec Security Association in the Security Associations Database SAD.
If there is no SA yet, the IPsec module sends a request to the IKE process to create an SA, so a key-exchange with the host Bob.
The IKE process negotiates keys and crypto algorithms with the peer host using the IKE protocol, then the SA is created and inserted in the SAD database.
Finally, the IPsec module can send the packet with applied IPsec.

XIV. SIEM (Security Information and Event Management)

- Introduction to SIEM

Those are special devices that are used in many big companies that try to have a single-point where you can have an overview of overall state of the system.
If you image a big network, composed by several subnets and thousands of hosts, you can have a privileged place where is possible to observe all activities within the network.
SIEM is supposed to combine together Security Information Management (SIM) and Security Event Management (SEM).
SIM is related to collecting log data for analysis and alerting security responsible; SEM realizes the real-time system, notifies network admin of important issues and try to establish event correlation.
The event correlation establishment is an intelligent evaluation of what's happening in the network, so that you can try to derive final conclusions of what is really happening in the network.

Typically, a SIEM, is generally made by multiple monitoring and analysis components; it is not a single tool or app, but a set of different building blocks; it meant to help organizations detect and mitigate threats.
The idea is that to combine together different tools, collecting different kinds of data, sending these data to the collector, the collector tries to put together the data and correlate them with engine and intelligent mechanism, and try to help the network administrator that something is happening in the network.
It also provides access to different kind of statistics and analysis, or also access to extra place of the system.
To have a very powerful SIEM we have not to consider only the components that made it, but also the data we are collecting and sending to these components.
An important problem is that SIEM, because it is made by the cooperation of several different components, is very complex and needs very skilled experts to be used.

There is no SIEM standard protocol or established methodology, but there are some common characteristics of SIEM systems:

- *automatically collect and process information from distributed sources*: imagine several different hosts in the network, there is an agent that provides data to the SIEM.

- *store data in one centralized location*
- *correlate between different events*
- *produce alerts and reports based on this information*: reports are important for the regulation, the compliance of the GDPR.
- *help for compliance and security incident management (digital forensic)*

The SIEM could be agent-based or agentless but is more common to have SIEM agent-based: there is a software placed in the source from which we want to take the data, and the configured to send these data to the SIEM.

The services provided by the SIEM are:

- *Log management*
- *IT regulatory compliance*
- *Event correlation*
- *Active response*: trigger response to some threat that is happening
- *Endpoint security*: all the hosts that have the agent installed can push to the log all the patches they've installed, like for example an apt packet manager logs. In this way it is possible to see if some host must update some version of some software, and trigger an update on a specific machine.

A log is not an event, an event is a bit more complex than one single line/string.

Log management

Log management is based on raw data on top of which you can build the security of the SIEM.
Nodes in an IT system, the more critical nodes, send relevant system and application events (logs) to a centralized database that is managed by the SIEM.
The most used logging solutions are Syslog, Syslog-NG, Splunk, LogStash and Graylog.
This SIEM database application first parses and normalizes the data sent by the numerous and very different types of nodes in the IT system.
Then the SIEM provides log storage, organization, retrieval and archival services.

The logs are the base, on top of the raw data the analysis is built.
Logs of different kind are correlated all together to observe from different point of view the same event.
The more nodes that feed into your SIEM system, the more complete and accurate your vision is of the IT system.

- IT Regulatory Compliance

Once the logs are stored, it is possible to build filter or rules and timers to audit and validate the compliance, or to identify violations of compliance requirements.
It is a prove of what's happen, for example proving that you are collecting and storing data needed by the regulation.
Other examples: monitoring frequency of password changes, identifying OS and applications patches, IDS updates.
SIEM can automatically produce reports often needed by business to provide evidence of self-auditing and to validate their level of compliance.

- Event correlation

This is the idea to combine together different elements so that you can find and deduce something happens considering different evidences.
The correlation engine on a SIEM can investigate and correlate other events that are not necessarily homogeneous.
It can provide a more complete picture of the health status of the system to rule out specific theories on the cause of given events.

- Active response

Activate procedures after the identification of given events: automatic response or manual response.
The SIEM triggered, automated, and active response to the perceived threat would probably occur much faster like adding IP and port filter on the ACL on a router or firewall.

- Endpoint Security

Most SIEM systems can monitor endpoint security to centrally validate the security "health" of a system.
SIEM systems can even manage endpoint security, by making adjustments and improvements on the remote system: configuring firewalls, download and install updates, adjust the ACL on a misconfigured personal firewall.

- ## Logs

Logs are the events that the network produces, without them is not possible to achieve any security management.
The typical problems related to logs are:

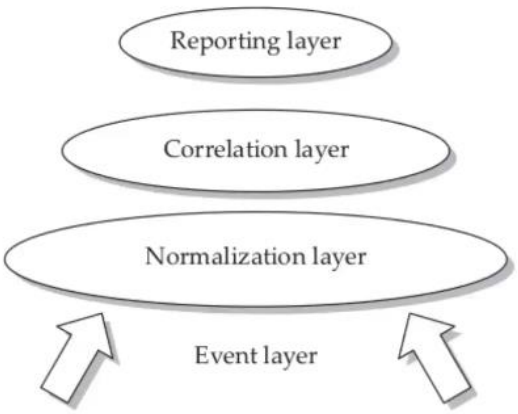
- *period/time of logs retention and data destruction*: typically this is based on the country regulation or guidelines of the company related to the data storing and destruction, like sensitive data of users.
- *type/kind of information system log*
- *store of the logs:* local storage, or cloud or hybrid solution, related on the security and the increasing amount of data we can have.

Log sources could be almost everything, but we have to take into account only the critical and important logs.
The decision depends on which we want to monitor: which devices (critical servers, firewall, IDS, hosts), which events we are interested on (debug info, configuration changes, log-in records, alerts).

- ## Event data vs State data

Event data provide you with an exact list of all events that occurred on your server, network or website, it is asynchronous and unexpected.
State data is a picture that gives you the view of the overall state of the system, it is synchronous: configurations of firewall, current running applications, active users, processes, registry settings, vulnerabilities.

- ## SIEM stack



The lower layer is the Event Layer where you collect raw data and data related to events.
Then, the first layer you find is the Normalization Layer because different hosts can generate different types of logs, or different application can generate logs with different structures and semantics.
The Correlation Layer tries to combine together different events in order to provide a picture something more interesting.

Finally the Reporting Layer, when the user queries the system for knowing the situation.

- Log correlation

Log correlation means monitoring the incoming logs for logical sequences, patterns, relationship, and values.
The ultimate goal is to analyze and identify events invisible to individual systems: it is not something that happens on a single source of log, but the idea is to be combining together different sources to realize that something critical is happening.

- Other concepts

SIEM supporting data is the concept to add to the original data some other information that are taken from other sources like names, ip addresses, os, software versions, geo info.

Event correlation is related to a special engine that is typical of different SIEM and it is the intelligence that the vendor sells.
Before event correlation they have to perform event normalization.
The original event correlation are very simple rules that trigger alerts or actions, now the machine learning is the most used one.
There is a big similarity in the concept between Event Correlation System and IDS, the difference is that with event correlation we are considering logs, while with IDS we are analyzing the network traffic.

Endpoint security is related to patch the OS and applications, updates of antivirus or antispyware, checking that firewall is on and configured properly, and also realize some changing in the rules; HIDS and HIPS, management of removable media (USB drives, CD, DVD), Network Access Control (NAC), Network Intrusion Protection System (NIPS).

IT regulatory compliance is related to all form of compliance related to the fact that “have you taken the steps to perform your responsibilities to secure manage the information in your control? If something happens, can you prove and have evidences that you have done all the security measures to protect the sensitive data and to perform your duties?”. This is for example related of GDPR.
In addition, we can **provide evidences of the best practices**.
Implementing technologies to protect and detect intrusions is not enough, this should be provable.
For example a log server must be reliable, it must use TCP protocol and encrypted

storage. Also it is important that the logs are not corrupted or manipulated, so we have to give them authentication and integrity.
For this reason, it is possible that SIEM can include a compliance checklist, that is a form that contains all the needed steps for adapting to the regulation.

Additional features:
- *Support for open-source threat intelligence feeds*: for example, the logs have only the IP address, the idea is to perform for example a whois-query for every IP and provides the domain related to the IP.
- Real-time analysis and alert
- Automated response
- Advanced search capabilities: elastic search
- Historical and forensic analysis

Threat intelligence is used helps organizations in taking security decisions: if something follows a pattern, then it is very likely that something is happening in the network. If something is detected, you can take some countermeasures in advance to mitigate possible damages.

Digital forensics
● Digital forensic science is a branch of forensic science focused on the recovery and investigation of material found in digital devices and cybercrimes
● Digital forensics was originally used as a synonym for computer forensics but has expanded to cover the investigation of all devices that store digital data
● Digital forensics is concerned with the identification, preservation, examination and analysis of digital evidence, using scientifically accepted and validated processes, to be used in and outside of a court of law

SIEM operational interfaces
● *Dashboards and maps → pull of information*: a way to present information in a way that administrators can understand at glance.
It is a graphical and organized representation of alerts, event data, and statistical information. It allows administrators to see patterns, understand trends, identify unusual activity. Dashboards are also key differentiator among the different SIEM products on the market
● *Alerts → push of information*: Do not require human diligence to notice something important is happening

SIEM is very complex system:

A survey conducted in 2013 by elQnetworks revealed that:

– “managing the complexity of the product is considered the biggest headache when using SIEM

– followed by lack of trained personnel to manage the product and lack of integration with other products”

– 31 percent would prefer to replace their existing SIEM solution for better cost savings

– 25 percent have invested more than month in professional services since the implementation of their current SIEM solution

– 52 percent require two or more full-time employees to manage the chosen solution

XV. Intrusion Detection Systems

- Intrusion detection and prevention systems

An **Intrusion Detection System (IDS)** is the idea of having a device that is able to monitor the network, try to understand what's happening in the network and to detect if something suspicious is happening (presence of intruders) before serious damage is done. Usually, we call it detection of an "anomaly" in the system.

The ultimate purpose of an intruder could be to:
- prevent legitimate users from using the system (DoS attack)
- reveal confidential and sensitive information, stealing data
- use the system as a stepping stone to attack other systems

Second generation IDS are **Intrusion Prevention Systems (IPS),** also produce responses to suspicious activities, for example, by modifying firewall rules or blocking switches port.
We already mentioned this kind of activity related to endpoint security using the SIEM.

In network IDS, it tries to do the **deep packet inspection**: inspect the payload of the packet that is moving in the network, trying to figure out signatures of viruses or malware, strange binary sequences typical of executable ransomware, and so on. The IDS has the task to report the intrusion; the IPS try to block the intrusions. Usually the IDS and IPS is placed right after the firewall.

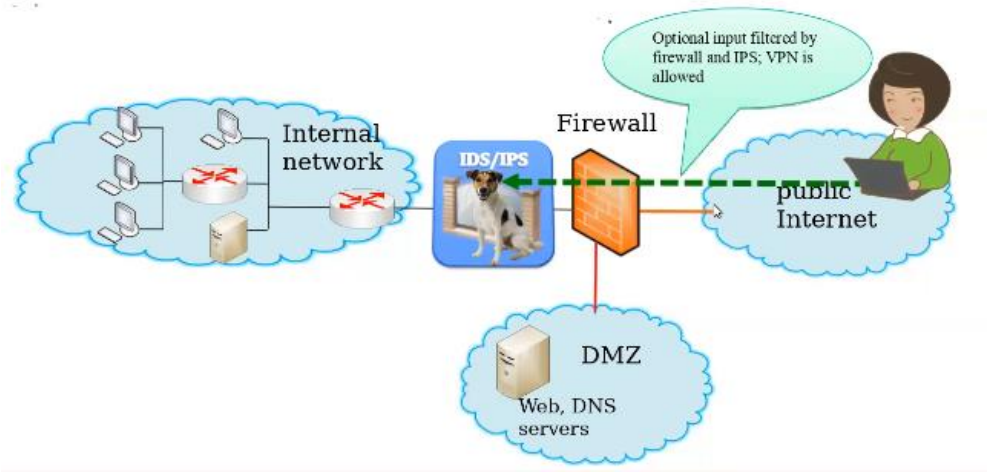


Figure 104: Example of a network with IDS

It is possible to set the IDS before the firewall but is not so useful because it will inspect a lot of traffic that will be blocked by the firewall.

There are several different IDS in different networks; the IDS are configured in different way according to the different networks that are monitored.

The IDS is usually **out of band**: this means that it is not really an hope into the network; it is a concept very similar to the monitoring used with the SPAN port and tap device in the switch. IDS is passive, it cannot really block the traffic, it can detect and raise alarms.
The IPS instead is usually **in line**: IPS is active, it is one of the node of the network and can decide to make a packet to be dropped.
The IPS detects and reacts by blocking intrusions in real time: drop packet, change firewall rules, and so on.
The NIST institute uses the term for them of IDP (Intrusion Detection Prevention) system.

The IDS is a very heavy process, then is the need of a very powerful machine to realize it. This fact depends on the processing the data and on the amount of traffic that is analyzed.

- Alarms

The concept of quality of IDS is very important.
It is related for example to the use of machine learning for classification, to decide if a sample belongs to a class or another class.
This is a concept that could be also apply in a parallel way to IDS.
There is the event packet received, and the IDS has to decide if the packet is legitimate or if it is an intrusion.
If the event “bad packet received” happens, then the outcome of IDS could be “it is an intrusion” or “it is a good packet, is not an intrusion”.
In this scenario we can distinguish between a **true positive** and a **false positive**.
The true positive is when the system receives a “bad packet” and the IDS confirm that it is a tentative of intrusion. The false positive is when the system receives a “legitimate packet” and the IDS wrongly alerts that it is a tentative of intrusion.
On the other side we can distinguish between **true negative** and a **false negative**.
The true negative is when the system receives a “legitimate packet” and the IDS confirm that it is normal traffic. The false negative is when the system receives a “bad packet” and the IDS wrongly said that it is normal traffic.
The ideal IDS would have that there are no false positive and no false negatives, only true positives and true negatives, so it is never wrong.

- ## IDS Functionalities and block diagram

Recording information related to observed events.
Notifying alert security administrators of important events.
Producing reports using details on particular events of interests.
In case of IPS changing the network activity: drop connections, block accesses, change configurations of devices, change content of packets.
Very similar to the SIEM.

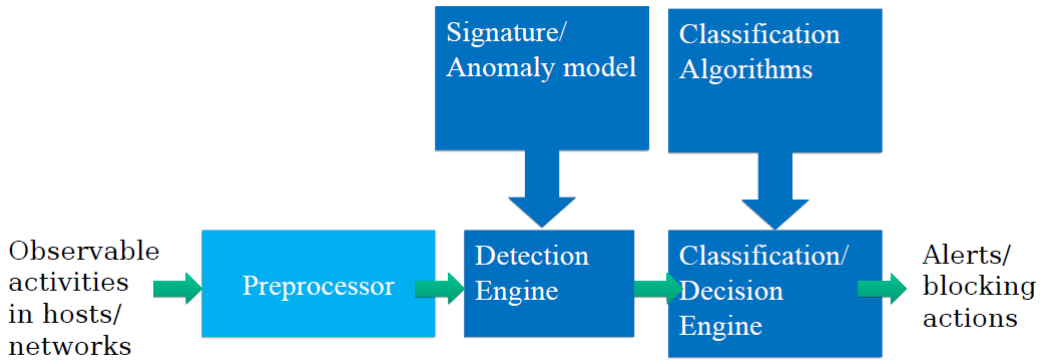


Figure 105: IDS/IPS block diagram

1) The starting point is to collect the observable activities (raw data) in hosts and networks.
2) Then there is a *Preprocessing* phase in which there is a normalization of the data or to focus on only a subset of the features that we could take from the data. This last activity is a function called *Feature Extraction*.
3) Then there is the core of the IDS is the *Detection Engine*: usually DetEng uses a *Knowledge Base*, that is a reference model as a comparator to take the decisions. According to the different kinds of model we can have different types of IDS: signature-based IDS, behavior-based IDS, and so on.
4) After the DetEng phase, there is the final step represented by the *Classification and Decision Engine* that uses different type of classification algorithms.
Finally we can have blocks or alerts according to the type of the IDS.

- ## Activities monitored

The activities monitored by IDS/IPS must be any activity sensitive to occurrences of any events deemed to be security concerns.
Between them, we can have:
- **Attempted breach**: reconnaissance activities (network scan, dir list scanning, user enumeration), patterns of specific commands in application sessions (login

and location frequency), content types with different fields of application protocols, network packet patterns between protected server and the Internet (model the ideal interaction and try to figure out if the real pattern is not admitted), privilege escalation.
- **Attacks by legitimate users:** illegitimate use of root privileges, unauthorized access to resources and data, command and program execution (file, database access, system calls).
- **Malware**: rootkits, trojan horses, spywares, viruses, zombie, worms, scripts.
- **Denial of service attacks**

The debate about how to classify the activities of monitoring is really open field of research in the sense to decide and select which elements could be used to detect a specific threat.
For example, ransomware is a program that encrypts the files, so there is a burst of activities related to the reading of the files and the writing of the data (encrypted). In this way the IDS could identify a ransomware, but this is one of the many methods that could be used.

- Types of IDS

Host-based HIDS
It is a small software installed in the endpoint that monitors the resources and events in the single host to detect suspicious activity.
It is typically used on critical hosts offering public services.
fail2ban is a software that is used as HIDS: if something fails too much, then it is possible to block this possible threat.

Network-based NIDS
The NIDS monitors the traffic exchange by different hosts.
It analyses the network, transport and application protocol activity.
It is placed behind a router or a firewall, just right after the access to the internal network.
Advantages: NIDS can protect many hosts and detect global patterns

Wireless WIDS
The WIDS analyses wireless networking protocol activity.
So it operates on the radio field. It is deployed in or near an organization’s wireless network.

- HIDS

The HIDS only monitors traffic on one specific system endpoint.
Usually does not need the promiscuous mode because of the single host.
It looks for unusual events or patterns that indicate problems: unauthorized access and activities, unexpected activity, change in configurations, software changes.

In Windows systems, the configurations files have typically a digest of them in another folder, and periodically it compares the original configuration with the digest.
So if there is any deviation, then it detects that something suspicious is happening.

- NIDS

The NIDS usually operates in promiscuous mode, like a sniffer.
It is like a Wireshark monitor that not only show the data but also to understand that and try to recognize them.
The idea is to use a database of previous intrusion patterns, and when a traffic that follow and is similar to one of these patterns is detected, it is possible to raise an alarm.
The NIDS is usually connected to switches with ports mirrored, or in SPAN mode (Switch Port ANalizer): that means all the traffic generated within all ports of the switches are replicated on the mirrored port where the NIDS is placed.
Often it has a series of sensors or extra devices placed in different networks like DMZ, internal net or specific nodes and that try to send related information to the monitoring node: we usually call it a Distributed Detection System.

- Other types of IDS

- *File integrity monitors (Tripwire)*: monitor the changes of the key system configurations files.
- *Flow-based IDS (NetFlow)*: instead of having a full deep packet inspection, it tries to summarize the exchange between different hosts using the notion of flow.
The flow is a summary of the exchange that happens with different hosts: average, total packets, total bytes/second, and so on.
The Flow-based IDS is very efficient in terms of resources used to make detection; they don’t look at the data but at statistics of the packets, so they use a kind of statistics-detection.
It is based however on building the typical model of interaction and flow between the hosts and the server, and then if there is a flow that deviates from this ideal

interaction than it is possible to raise alarm. For example if a port is not used typically in the network, then if there is an exchange on that port it is suspicious.
- *Hybrid Detection Capabilities*: it is usually anomaly behavior-based and often require training periods to establish a baseline.

- IDS approaches

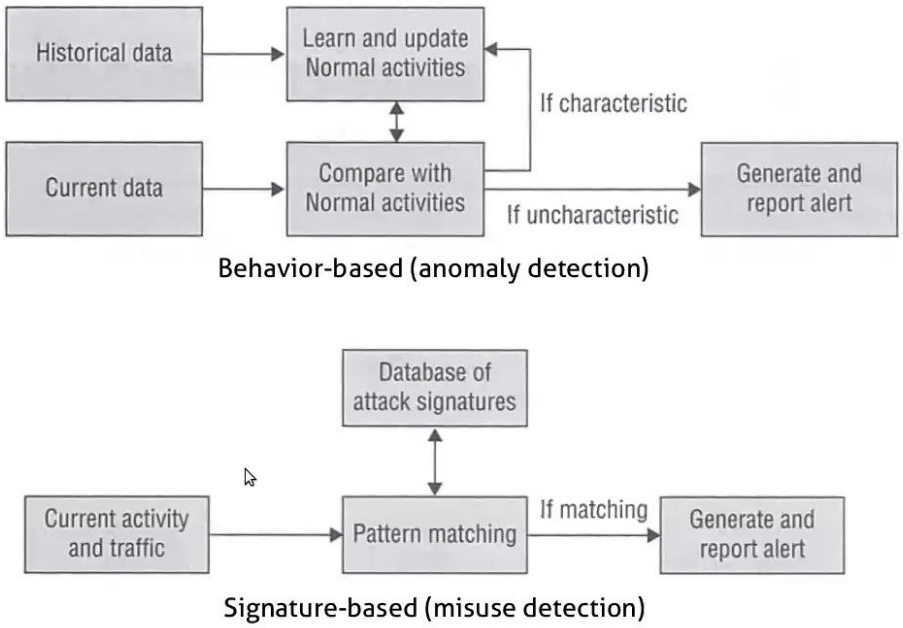


Figure 106: IDS approaches, behavior-based and signature-based

There are two approaches: the **behavior-based (anomaly detection)** and the **signature-based (misuse detection)**.
In the behavior-based, we use a statistical approach by taking historical data that are used to learn and update the model of standard behavior that we consider the normal activities; this standard model is used with the Detection Engine that compares the current data with the standard model and try to see similarities.
If the current data does not follow this model of the path, then an alert is generated.
In the signature-based, the IDS uses a pattern matching Detection Engine that uses a database of attack signatures: a signature in this context can be considered a very typical characteristic of an attack.
For example, the worm Blaster was composed by one single small UPD packet that was using the remote procedure call of Windows OS.
This packet has very specific characteristics that makes it very easy to match the signature.
As before, if there is a match then a report is generated.

The behavior-based approach is not so much precise, and it is possible that something strange happens: if for example there is some failure in the cable, then the packet maybe is sent many times and is considered as a DoS, so a false positive. It is very difficult to define all possible normal activities.
In this case the number of false positives is increased.

Using the signature-based approach and the related database, the number of false positives is reduced: if a packet matches the signature, is very likely that is an attack. The disadvantages are that in this scenario the number of false negatives is increased because it can only detect the attacks that are matching the entries in the database; if an attack is not in the database or is a modified version, then it is possible that is not reported as an attack.

The behavior-based define characteristics of normal behavior, the signature-based define characteristics of suspicious and abnormal behaviors.

- Learn and classify anomalies

Anomaly detections requires some form of **learning or training**: it is usually based on data mining in actual observations, takes from the data the relevant part.
The typical problem of this kind of learning tasks is to have a big amount of data that are complete.
The features fall into three classes:
- Intrinsic features: data about a particular connection or flow
- Traffic feature: statistical information about connections
- Content features: application-related statistics (very specific)

- Behavior-based IDS

Intruders may behave in a different manner from ordinary users and programs, many types of attack are characterized by abnormal patterns of network use. Recognizing abnormal behavior enables us to detect attacks as they take place.

One of the most important things is just to use the observed data (system calls, network elements, and so on) to build a model and try to define activities.
It is possible to derive the “normal” behavior, generated using statistics (like distance measures and thresholds, the Hamming distance) or with a set of rules (like parameters) or with a Machine Learning approach (probability-based approach using Markov model, Neural networks and so on).

Usually, the traffic is never static but is dynamic and can continuously change. Then self-learning is critical to ensure wide and successful deployment of anomaly-based detection mechanism. The model should adapt to the changes of the network.

- Signature-based IDS

Signature-based IDS is older than the behavior-based one.
It starts from the idea that intruders may have a characteristic appearance which makes it possible to identify them.
The idea is to screen the payloads of the packets looking for specific attacking patterns called signatures.
Suppliers of IDSs maintain huge databases of signatures (code or data fragments) which characterize various classes of intruders.
If the goal is to have a good rate of detection, then the IDS must have rapid recognition, that means searching for matches for one or more of the known signatures from a collection of many thousands.

The signature-based rule-based IDS is the dominant technology in the commercial systems.
Rules express actions on given conditions, including protocol, source, destination, timing and payload content.
It practically is a powerful “packet-sniffer” that captures the packets in a LAN and looks for specific patterns into the payloads. Rules can be derived from more automatic IDS.
SNORT is one of the most known signature-based IDS and Suricate is its evolution.

IDS cannot inspect encrypted traffic like VPNs and SSL; the payload is encrypted and cannot be accessible by the IDS. Not all attacks come from the Internet.
Also another problem is that, according to the magnitude of the network, IDS has to record and process huge amount of traffic.

- Misuse detection

Set of rules defining a behavioral signature likely to be associated with attack of a certain type.
Example:
- the buffer overflow: a setuid program spawns a shell with certain arguments, a network packet with a lost of NOPs in it, a very long argument to a string function.
- syn flooding (DoS): large numbers of SYN packets without ACKs coming back.

The problem is that attack signatures are usually very specific and may miss some modified versions or variants of the same attack.
The signatures must use invariant characteristics of known attacks, like port number of app with known buffer overflow, bodies of known viruses, but this is hard to achieve with malware mutations, like metamorphic virus.

The **honeypots** are useful for signature extraction, they attract attackers and malicious activity so that it is possible to study the attack.
It is a security resource whose value lies in it being attacked or compromised.
A honeypot is typical a single computer or honeynet is a network of computers.

- IDS rule evasion

Example of scenario.
The IDS has a rule to detect “USER root” in a packet stream.
Scanning for it in every packet is not enough: attack can split attack string into several packets, this will defeat the NIDS.
Recording previously packet’s text is not enough because attackers may send packets out of order.
Full reassembly of TCP state is not enough, the attacker can use TCP tricks so that certain packets are seen by NIDS but dropped by the receiving application: manipulate checksums, TTL, fragmentation.

TCP Attacks on NIDS – The Insertion Attack

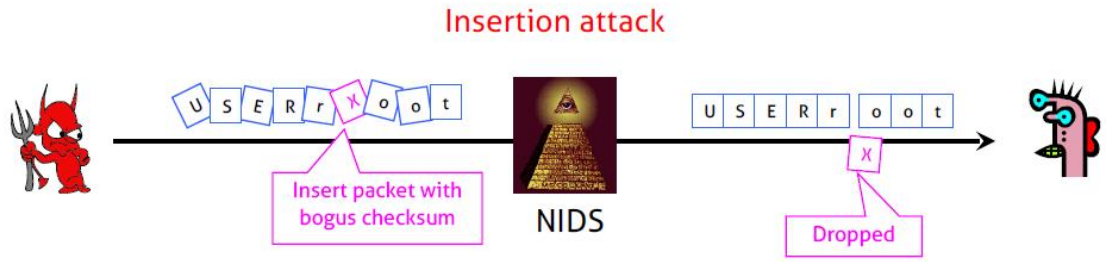


Figure 107: Insertion Attack

The idea for this attack is to insert into the payload the attacker add another set of bytes that have a bogus checksum, that means it is a corrupted checksum.
Then it happens that the IDS will check only the sequence without verifying the checksum (how much processing must be done by the IDS on an huge amount of traffic to be fast?), but when the packets reach the destination host, the packet with corrupted checksum is dropped, and the attack string is recreated on the host.

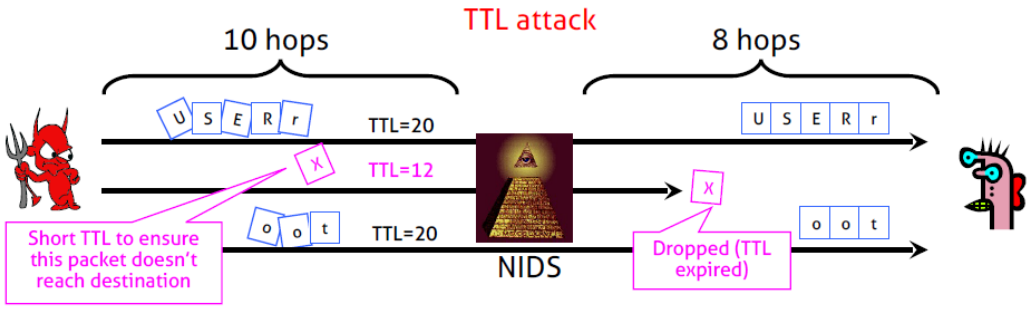


Figure 108: TimeToLive Attack

In the TTL attack, all the packets have a TTL that is enough to reach the destination except the one that we want to be lost, then it has a lower TTL to ensure that it does not reach the destination because TTL expired.

- ## Signature-based vs Behavior-based

Signature-based detection clearly indicates the detected attack method; on the other side the behavior-based detection indicated the attack type, the behavioral rule that was violated (like port scan), the statistical profile that was violated.
The behavioral protection can not identify the specific class of attacks or exploit that was blocked: this could be useful for detecting new kind of threats never seen before.

The false negatives, when the attack is not detected, is a problem in signature-based; the false positive is a problem in statistical anomaly detection in the behavior-based. Then also the signature-based cannot detect Zero-day exploit, DDoS, protocol anomaly.
The best option is to **combine behavior and signature based IDS**.
By combining them, it is possible to have a detection correlation, and once an exploit has been recognized using the behavior-based technique, a stateful signature can be created to provide accurate detection.
In this way it is possible to lower the false positives rates and reduce the response time to attacks.

- ## IDS, IPS, Observations

IDS is out of band.
An IDS false positive is an alert that did not result in an intrusion; it does not interfere with the traffic.
IPS is in band, and need higher hardware requirements.
When an IPS has a false positive, legitimate traffic will be blocked; so IPS should really limit the rate of false positives.

Legitimate traffic in real networks contain anomalies that can come from exceptional, but critical, business processes for example.
The IDS filters create leads on this suspicious activity but this need an expert to follow. Anomaly-based detection mechanisms are useful for IDS (for alerts) but not appropriate for IPS (as said with false positive, legitimate traffic is blocked).

The IDS architecture is composed by several parts, in a distributed system:

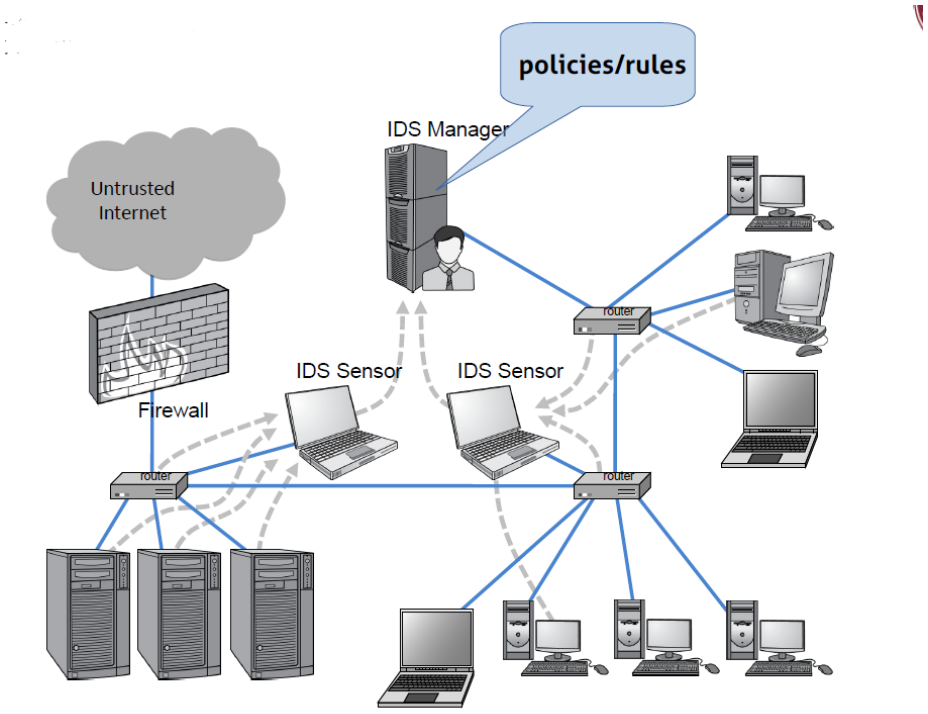


Figure 109: IDS Architecture

- network sensors: detect and data to the system
- central monitoring system: a server that processes and analyzes data sent from sensors
- database and storage component: repository for event information

Laboratory Activities

- Configure manually IP Addresses

Add IP address 10.0.0.1 with netmask /8 to eth0 interface

- ip address add 10.0.0.1/8 dev eth0
- ifconfig eth0 10.0.0.1 netmask 255.0.0.0.0

Add IP address 192.168.100.26 with netmask /29 to eth0 interface and broadcast address:

- ip address add 192.168.100.26/29 broadcast 192.168.100.31 dev eth0

Delete IP address 10.0.0.1 with netmask /8 from eth0 interface:

- ip address del 10.0.0.1/8 dev eth0

Delete any IP Address from eth0 interface:

- ip address flush dev eth0

Show all IP addresses of all interfaces:

- ip address show

Show IP address of interface eth0:

- ip address show eth0
- ifconfig eth0

Show interfaces:

- ip link show

Bringing interface eth0 up/down:

- ip link set eth0 up
- ip link set eth0 down

Set MAC address:

- ip link set eth0 address 00:11:22:33:44:55

- ## Routing Table

Show the routing table:

- route
- ip route list

Add default gateway:

- route add default gw 192.168.100.30

Add next-hop routing rule:

- ip route add 10.0.0.0/8 via 10.0.0.1

Del next-hop routing rule:

- ip route del 10.0.0.0/8 via 10.0.0.1

Add default routing rule:

- ip route add default via 10.0.0.1

Add direct forwarding rule:

- ip route add 10.0.0.0/24 dev eth0

- ## Configure IP Address via interfaces eth0

It is possible to create the file */etc/network/interfaces.d/eth0* with this content:

```
auto eth0
iface eth0 inet static
   address 192.168.100.25
   netmask 255.255.255.248
   gateway 192.168.100.30
```

Remember to load the configuration by using the command: *ifup eth0*

- ## Setup the DNS via /etc/resolv.conf

It is possible to edit the file */etc/resolv.conf* with this (for example) content:

```
nameserver 8.8.8.8
```

- Setup the DHCP via /etc/udhcpd.conf

It is possible to edit the file */etc/udhcpd.conf* in this way (for example) by imaging a network at 192.168.100.16/28 with the router at address 192.168.100.30:

```
# The start and end of the IP lease block
start           192.168.100.17      #default: 192.168.0.20
end             192.168.100.29      #default: 192.168.0.254

# The interface that udhcpd will use
interface eth0          #default: eth0

# The maximim number of leases (includes addressesd reserved
# by OFFER's, DECLINE's, and ARP conficts
max_leases          13          #default: 254
```

Then at the end of the file look at the options:

```
opt     dns     8.8.8.8
option  subnet 255.255.255.248
opt     router 192.168.10.30
option  domain pnd.test
option  lease  600
```

To start DHCP serviced on the server terminal type:

- udhcpd -f /etc/udhcpd.conf

On the client terminal instead:

- dhclient ethX

- Capturing packets using tcpdump

Capturing packets on the interface eth1:

- tcpdump -i eth1

Capturing only IPv6 packets:

- tcpdump ip6

Removing timestamps:

- tcpdump -t

Verbose options (extra details):

- tcpdump -v
- tcpdump -vvv

Save an exchange of packets captured with tcpdump and then transfer outside Kathara:

- tcpdump -w result.pcap
- cp result.pcap /shared/

- Capturing packets using Wireshark

- IPv6 Addresses

Scenario: LAN1 and LAN2 must have the subnets **2001:DB8:CAFE:1::/64** and **2001:DB8:CAFE:2::/64** respectively.
Hosts pc1, pc2, pc3 and pc4 must have interface id: 101, 102, 103 and 104. Router r1 has always 1 in its Interface ID, in Link Local and GUA.

Set GUA IPv6 of host PC1 to interface eth0:

- ip addr add 2001:db8:cafe:1::101/64 dev eth0

Add default route through router to the IPv6 hosts routing table:

- ip route add default via *router_link_local_IPV6* dev eth0

Show IPv6 routing table:

- ip -6 route
- route -6

Set Link-Local address of the router on both interfaces:

- ip addr add fe80::1 dev eth0
- ip addr add fe80::1 dev eth1

Set GUA address of the router on both interfaces:

- ip addr add 2001:db8:cafe:1::1 dev eth0
- ip addr add 2001:db8:cafe:2::1 dev eth1

Add Networks route through interfaces in the router:

- ip route add 2001:db8:cafe:1::/64 dev eth0
- ip route add 2001:db8:cafe:2::/64 dev eth1

Show IPv6 neighbors devices (with associated MAC Addresses):

- ip neigh

- Configuring IPv6 through /etc/network/interfaces.d/ethX

Example of configuration of host PC4 via */etc/network/interfaces.d/eth0:*

```
auto eth0
iface eth0 inet6 static
   address 2001:db8:cafe:2::104
   netmask 64
   gateway fe80::1
```

Remember to: *ifup eth0*

- SLAAC configuration in Kathara

System Control (sysctl)

Run in privileged mode to change settings.

See all options: *sysctl -a*
Search IPv6 settings in the net section: *sysctl -r ipv6 net*
Change a variable:

- *sysctl -w variable_name=value*
- *sysctl -w net.ipv6.conf.eth0.accept_dad=1*

See a variable value:

- *sysctl variable*
- *sysctl net.ipv6.conf.eth0.accept_dad*

- Generate EUI-64 Algorithm Interface ID

1) Configure interfaces file of the host /etc/network/interfaces.d/eth0
auto eth0
iface eth0 inet6 auto
2) Run the Router Advertisement Daemon: in the router terminal type *radvd -n*
3) Down and up the eth0 link on the host:

- *ifdown eth0*
- *ifup eth0*

Host has received the **GUA Address (with EUI-64 Algorithm).**
The GUA Address is generated by default using the EUI-64 Algorithm.

- Generate Random Interface ID

1) Configure interfaces file of the host /etc/network/interfaces.d/eth0
auto eth0
iface eth0 inet6 auto
2) Set the option in the host:
sysctl -w net.ipv6.conf.eth0.addr_gen_mode=3
3) Run the Router Advertisement Daemon, in the router terminal (if not started yet):

- *radvd -n*

4) Down and up the eth0 link on the host:

- *ifdown eth0*
- *ifup eth0*

Host has received the **GUA Address with Random ID**.
The Random ID generated for the host, if we do it again, will be the same because it depends on the secret and on the prefix of the network.
This is defined in the RFC217, in the ***stable_secret*** parameter.

- Generate Temporary Address

1) Configure interfaces file of the host /etc/network/interfaces.d/eth0
auto eth0
iface eth0 inet6 auto
2) Set the option in the host:
   - *sysctl -w net.ipv6.conf.eth0.addr_gen_mode=3*
   - *sysctl -w net.ipv6.conf.eth0.use_tempaddr=1*
   - *sysctl -w net.ipv6.conf.eth0.temp_valid_lft=180*
   - *sysctl -w net.ipv6.conf.eth0.temp_prefered_lft=60*
3) Run the Router Advertisement Daemon, in the router terminal (if not started yet):
   - *radvd -n*
4) Down and up the eth0 link on the host:
   - *ifdown eth0*
   - *ifup eth0*

Host has received the **GUA Address with Random ID** and also the **temporary addresses** that are generated every short amount of time.

- OPNsense

OPNsense is an open-source router-firewall based on a particularly robust version of BSD. The default behavior is to DENY all.

- Built-in intrusion detection and prevention system (Suricata)
- Availability of the most common network services (DNS, DHCP, captive portal, proxy, traffic shaper).
- Management of different types of VPN
- Management of backups and backups recovery
- Possibility of expansion through plugin

To access the OPNsense control panel, just enter the IP address in the web browser – ex: 100.100.4.1
In the login window, enter root as user name and opnsense as password

Management Plane Opnsense

● **Change admin user password**

– Lobby: Password

– Enter a password other than opnsense

– Save the new password → SAVE

● **Make sure to log in only using HTTPS**

– System: Settings: Administration → Protocol HTTPS

– Apply the new settings → SAVE

Caution: it may be necessary to reload the page and authenticate in the firewall again for switching to encrypted communication

● **Configure NTP service on internal interfaces**

– Services: Network Time: General → select INTERNAL, DMZ, EXTERNAL

– Save the new settings → SAVE

● **Send logs to a syslog server**

– System: Settings: Logging / targets → add (+)

– Add as target the machine logserver.acme-XX.test, listening on port 514 via UDP.

To limit the number of logs, do NOT send messages for debug and info levels.

– Save new settings → SAVE

– Apply changes → APPLY

Control plane Opnsense

1) Limit incoming ICMP traffic

Firewall: Shaper: Settings

- In Pipes tab, add a limitation to 10 Kbit/s without selecting mask
- In Rules tab, a rule for WAN about ICMP packets from each source and for each destination, using the limitation defined in Pipes tab as target
- Apply changes → APPLY

2) Filter ICMP redirect messages

Firewall: Rules: Floating → ADD

– Add a Block rule in the first position, for ICMP redirect packets from any source and for any destination on the DMZ and EXTERNAL interfaces.

– The rule must be of type "in" since packets enter the interface.

– If necessary, to change the position of a rule, check the box next to the rule and click on the arrow icon (<-) at the position where you want to move the rule

3) Filter outgoing ICMP unreachable messages
Firewall: Rules: WAN → ADD
– Add a Block rule in the first position, for ICMP packets of type Destination Unreachable with outbound direction from each source and to each destination
– To change the position of a rule, check the box next to the rule and click on the arrow icon (<-) at the position you want to move the rule to
– The rule must be of type "out" since the packets we want to block go out of the interface to reach the network connected to it
- Apply the new rules → APPLY CHANGES

Data Plane Protection Opnsense
Like many other firewalls, the data plane protects itself by inserting traffic filtering rules. In Opnsense, rules are either interface-specific or floating.

● Floating rules are considered before any other interface rules.
A floating rule of type "pass" source any destination any overrides any other rule, making firewall rules quite useless.

● The rules for an interface are either "in" or "out":
– "in", when packets are generated by the network to which the interface is connected, and they want to enter the interface (so "in" input) to reach a different network
– "out", when another network generates packets, reach an interface, and want to exit the interface (thus "out") to enter the network to which the interface is connected

1) Block packets with spoofed IP address: filter packets with IP addresses not coming from local networks
- Firewall: Rules: DMZ → ADD
- Add a pass rule for packets originating from the DMZ net, destined for any host and with any protocol. The rules must be of type "in" since the packets enter the interface from the DMZ net network
- Create similar rules with the necessary modifications for the other internal interfaces, i.e. INTERNAL, EXTERNAL_CLIENT
- Apply the new rules → APPLY CHANGES

2) Allow only packets that match the traffic expected on the network: allow access to the DMZ only to packets that require the services provided
● Firewall: Rules: DMZ

– To disable the rule that allows all IP packets coming from any source to reach any
destination (it goes from a default pass to deny)
– To disable a "pass" type rule you can click on the green icon
● Firewall: Rules: WAN → ADD
– Add a “pass” rule for packets destined for the host with the webserver on port 80, coming
from any source.
– Add a “pass” rule for packets destined via tcp to the host with proxyserver on port 3128 coming from any source.
– The rules must be of type "in" since the packets enter the interface from the WAN1 network.
● Create similar rules with the needed modifications for the other networks
● Apply the new rules → APPLY CHANGES

- Tunneling

The tunnel concept could be used in any kind of protocol.
The idea is to take the packets, encapsulate them in a way that can be routed in the intermediate network, and finally the packets are delivered and decapsulated.
Any application can use that interface without any need to change its code: we create a *virtual interface* in the host, and we tell the application to use that interface if it wants to send packets to the other end of the tunnel.
With this concept, application generates the payload, those data are copied in a location memory that is the virtual interface, then the VPN application takes the payload, manipulates and process them, and send it into the real interface.
Usually we distinguish between 2 types of universal drives: tun or tap.
tun encapsulate IP layer, tap encapsulate Ethernet layer.

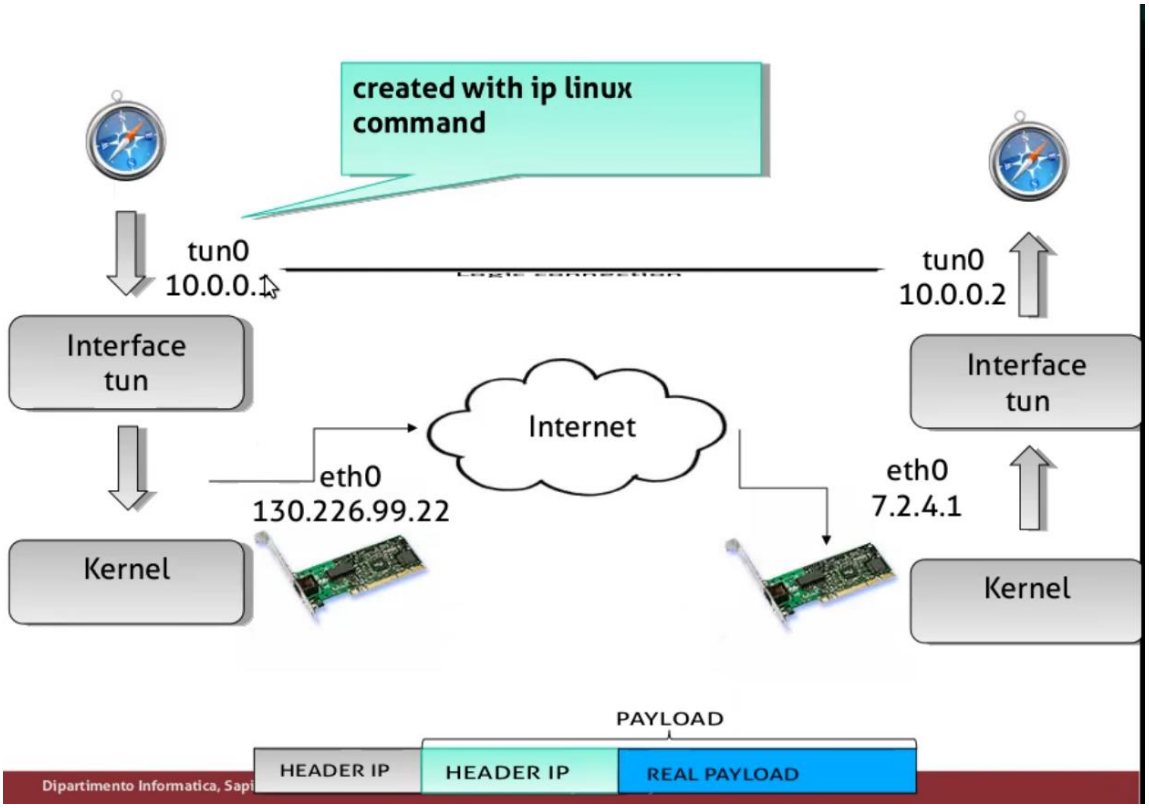


Figure 110: Example of tunnel

There is the creation of the virtual interface of tun0, that is an a logical connection with another virtual interface that is in another host.
Then, try to make a connection between the virtual interface tun0 and another interface that performs the real routing of the packet.
The packet that we sent into the tunnel, that is the Header IP + Real Payload, will be encapsulated in another packet with another Header IP, that is the one that will flow into the Internet. That is why we said IP over IP.

At the destination, Kernel recognizes that is IP over IP, decapsulate it, see that the destination is a local endpoint of the tunnel and forward it to the tunnel.

Commands:
- ip tunnel add tun0 mode ipip remote <remoteIPAddress> local <localIPAddress>
- ip link
- ip addr add 10.0.0.1/30 dev tun0
- ip link set tun0 up

- ## OpenVPN

OpenVPN is an open-source software to realize VPN, so encrypted tunnels.
It usually uses UDP with one single port but can also use TCP: this makes sense because it is possible to use a server that lists on the OpenVPN in the user space; if instead you want to works on lower-layer you must use to the Kernel layer. In this case you have IPsec that is practically already part of the Kernel and must be configured.
The use of UDP is preferred because if we use instead TCP in the tunnel, we are using a reliable protocol (TCP) on top of another reliable protocol (TCP) and this is not how the protocol is intended to work, a reliable protocol must be used on top of an unreliable protocol (IP).
The TCP protocol possibility was made for firewall reasons.
It can be used through firewalls or NAT, and it is based on the OpenSSL.
There are multiple modes: static or dynamic.

OpenVPN static key
The endpoints share a key generated with openvpn command
● Very easy to configure
● No CA or certificates
● NOTE: requires a secure channel to exchange the keys
● The key never changes: no forward secrecy

The keys that are generated are 4 and are independent: K_AB (to encrypt from A to B), HMAC_AB (Authentication from A to B), K_BA and HMAC_BA (parallel from B to A).

Generate the shared key on one side of the tunnel (say r1):
– openvpn --genkey --secret secret.key

● Exchange the secret.key file with scp
● OR, if you don’t send it with scp:
– Encrypt the key (because we'll use an insecure channel)
● openssl enc -aes-128-cbc -e -a -in secret.key -out secret.key.enc
– Exchange the shared key
● Prepare to receive the shared key on the other side of the tunnel (say r2):
– r2# nc -l -p 9000 > secret.key.enc
● Send the shared key
– r1# nc <r2IPaddress> -p 9000 < secret.key.enc
– Decrypt the key on the other side of the channel
● openssl enc -aes-128-cbc -d -a -in secret.key.enc -out secret.key

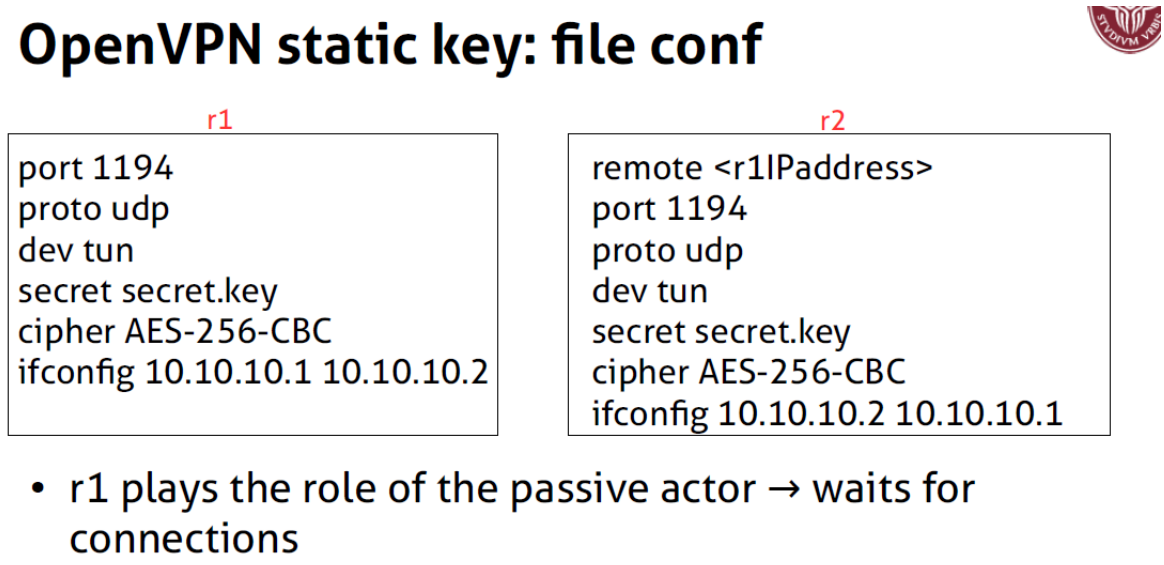


Figure 111: OpenVPN conf for static key

OpenVPN dynamic key

● Uses SSL/TLS and certificates for authentication and key exchange
● Certificates for both endpoints
● If the certificates are valid
– HMAC and encryption keys are dynamically generated with OpenSSL
– This assures Forward Secrecy
● Both parties contribute to key generation

We should have a certification authority issuing the certs
– This should be done using the → easy-rsa scripts, that can be found in the /usr/share/easy-rsa directory

● Alternatively, we can use the ones provided by openvpn for test purposes in /usr/share/doc/openvpn/examples/sample-keys
● Needed ingredients/files
– {client,server}.crt : CA signed public key
– {client,server}.key: CA signed private key
● Certificates are issued after signing the requests (client!=server)
– dh.pem: Diffie-Hellman key exchange parameters
– ta.key: for TLS HMAC authentication (optional)

- Filters in tcpdump e Wireshark