

Practical Network Defense notes

Matteo Salvino

Contents

1	Network 101	3
2	Traffic monitoring	5
3	IPv6 introduction	8
3.1	Notation and types	9
3.1.1	Unicast addresses	10
3.1.2	Multicast addresses	12
3.2	Addressing	13
3.2.1	Prefix Delegation process	15
3.3	Comparing the IPv4 and IPv6 headers	16
3.4	Link-local attacks	18
4	Internet firewall	23
4.1	Security strategies	24
4.2	Packet filters	25
4.3	Stateless firewall	28
4.4	RAW, MANGLE, NAT and FILTER tables	28
4.5	Stateful firewall	29
4.6	Other types of firewalls	30
5	Network Address Translation	31
6	Virtual Private Networks	33
6.1	SSL Tunneling	35
6.1.1	SSL/TLS Heartbeat	37
6.2	Device placement	38
6.3	IPsec	39

1 Network 101

Internet is in essence a network of networks with a hierarchical structure. The path into the Internet backbone could be wired or wireless. The backbone itself consists of global Internet Service Providers (ISPs) and several regional ISPs that are all interconnected to provide a path from sender to receiver. The communication path may typically contain a variety of switches and routers that facilitate and direct the flow of information through the network. The communication links, regardless of whether they are wired or wireless, are defined by a transmission rate and bandwidth. Access networks are used to connect a host or Local Area Network (LAN) to the Internet. Routers connect LANs, generate routing tables and forward packets of data on their path from source to destination. The Internet backbone, also called network core, is basically a group of routers interconnected by optical fiber as well as DNS servers containing infrastructure name servers, such as root Domain Name Servers (DNSs) employed for naming. The remaining components in the Internet structure that lie outside the network core, are simply access networks. An individual home network can be considered a small network or subnet. The Internet uses a gateway, also known as an edge router, to connect these networks to the edge of the network, also called network edge. The network core is composed of a set of routers and fiber links. The routers work together to determine the most efficient routing path for a packet from source to destination. A distributed algorithm is used that provides the flexibility to adapt to changing conditions, and routing tables are generated and maintained in real time. The core is provided by ISPs that interconnect multiple continents, called Global ISPs or Tier-1 ISPs, whereas the Regional ISPs are known as Tier-2 ISPs. The tier-1 ISPs that form the Internet backbone are interconnected at various access points called **Internet eXchange Points (IXPs)**. It would certainly appear that the intercommunication among computers would require some standardization that would facilitate their successful interactions. There should be some "protocol" that defines the manner in which they talk to each other so that messages are clearly understood. It is this "protocol", documented in a stack that is accomplished through modularization, development and upgrades that support operations such as web surfing, email, etc. By its very structure it is clear that the stack consists of different layers, each of which performs a special function. Modularization of the Internet is accomplished through layering. As a result, the Internet is being developed by many people, and institutions through a divide-and-conquer strategy. There is a strong interaction between layers in that each layer relies on the services of

the layer below and exports services to the layer above. It's the interface between layers that defines the interaction, for example implementation details can be hidden and layers can change without affecting other layers. Access layer is constituted by network with the same end-point, and the protocol used is Ethernet. Each host in a Ethernet network has a Network Interface Card (NIC) with a fixed address. MAC addresses uniquely identify hosts in the network and each of them processes packets intended for it. An Ethernet network constitutes a broadcast domain, where each Ethernet frame is received by all hosts. Actually, switches segment the network to limit the explosion of packets in the network, and only broadcast message are replicated. Switches remembers the source MAC addresses on the different ports. They only replicate the frame on the segment where the destination MAC address replies, with the support of ARP tables (for hosts) and CAM tables (for switches). Ethernet cannot be used for the entire Internet because of its broadcast packets, that would be highly inefficient for large networks. There is the need of a logical division of the networks : while Ethernet provides the Access layer, the Distribution layer is based on **Internet Protocol (IP)**, which is at level of Autonomous Systems (like big enterprises and ISPs). The main difference between Ethernet and IP address is that the first one is a physical addresses, so we cannot change the MAC address of our NICs, and tell who you are but not where you are, the second one is a logical address, so we can change the IP address of our NIC and is used to identify and reach networks and hosts. If two hosts belongs to the same network, then they can communicate using local addresses, otherwise they must use remote addresses. In order to know if one IP is in the same network than you, we can use the subnet mask. There are two version of IP addresses :

- **IPv4** : it defines IP address with 32 bits organized in four octets (8 bits in each)
- **IPv6** : has 128 bits.

For human readability the bits in each octet are separated by dots while writing an IPv4 address and colons in IPv6. IP addresses, regardless its version, are constituted by a network part and an host part, where the subnet mask defines precisely the boundary between them. There are three types of IP addresses :

- **Unicast** : these refer to a single destination host
- **Broadcast** : these refer to every host on a network or subnet

- **Multicast** : these refer to a group of IP addresses in a network, not necessarily all of them.

IP addresses has been classified in the following classes :

- Class A (24 bits for host addresses)
- Class B (16 bits for host addresses)
- Class C (8 bits for host addresses)
- Class D (multicast)
- Class E (reserved).

There are routable and non-routable address ranges. The first one need to be unique on the Internet (public), while the second ones (private, e.g. from 10.0.0.0 to 10.255.255.255, from 172.16.0.0 to 172.31.255.255, from 192.168.0.0 to 192.168.255.255) are defined in RFC 1918. Since each set of 8-bits can hold values from 0-255 we haven't flexibility at all. The idea is to use a Variable Length Subnet Mask in order to use a specific length of the subnet mask based on our needs. Many smaller networks can be grouped using the supernetting technique. In point-to-point links the subnet 31 is allowed. Other ways to reduce the waste of IP addresses in a subnet are NAT and IPv6.

2 Traffic monitoring

It's important to note that activities within the Internet can be approached in a modular fashion and this modularization is accomplished through layering. It is clear that the stack consists of different layers, each of which performs a special function. There is a strong interaction between layers in that each layer relies on the services of the layer below and exports services to the layer above. The task of a layer involves the exchange of messages that follow a set of rules defined by a protocol. When computers are connected with a network, guidelines must be established that support their interaction. The architecture that defines the network functionality is split into layers that collectively form what is commonly known as a protocol stack. Lets talk about **OSI model**. Each layer of the protocol stack may employ several protocols that implements the functionality of that particular layer. In a natural progression up the stack, the physical layer deals with the transmission of bits that are propagating over such media as copper, fiber

or radio. The data link layer aggregates the bits (e.g. in a frame), and performs the data transfer between neighboring network elements using as an example, Ethernet or WiFi. The network layer handles the routing of datagrams, in packet form, from source to destination using routing protocols. The transport layer performs the process-to-process communication using segments, i.e. message transfer using for example TCP. The session layer aggregates connections for efficiency, synchronization and recovery in data exchange. The presentation layer permits applications to deal with coding, encryption, and so on. Finally, we have the application layer which allow the user to invoke protocols for information exchange. Another well known model is **TCP/IP**. It's constituted by four layers : application layer corresponds to the top three layers of the OSI model, transport layer is equivalent to the transport layer of the OSI model, Internet layer is equivalent to the network layer of the OSI model, Datalink is equivalent to the data link layer of the OSI model and physical layer is equivalent to the physical layer of the OSI model. The layer ideal representation is that transport represent the illusion of direct end-to-end connection between processes in arbitrary systems, network transfers data between arbitrary nodes and data link transfers data between directly connected systems (via shared medium or direct cable). Each layer in the stack, with the exception of the physical layer, has a **header**. These headers facilitate the communication of information and are analogous to an envelope that contains both source and destination addresses. The link layer has a header containing MAC addresses, the network layer has a header containing IP addresses and the transport layer has a header containing the port number (i.e. service number). The port range is $[0..65535]$, where the source port is randomly chosen by the OS and the destination port determines the required service. The port range $[0..1023]$ contains well-known ports and they are used by servers for standard internet applications (e.g. 25 for SMTP, 80 for HTTP, 143 for IMAP). The ports in the range $[1024..49151]$ can be registered with Internet Application Naming Authority (IANA), whereas ports in the range $[49152..65535]$ are called ephemeral ports. Protocols are classified in two main categories :

- **connection oriented** (like TCP) : these protocols perform a number of very important functions. For example, they govern the movement of packets from source to destination under the specifications of certain standards, take actions that are specified in the packets, manage packet flow and congestion for optimal performance and even recover lost packets. The protocols work in conjunction with one another to accomplish the specified task required by the user.

- **connectionless** (like UDP) : these protocols doesn't provide any guarantee that the datagram is really deliver to the correct destination (i.e. there isn't control on data exchange). Furthermore, there isn't a possibility to recover from errors and manage the flow congestion.

The procedure to open a TCP connection between two hosts, A and B, is called three-way handshake, which is constituted by the following steps :

1. A sends a SYN segment to B with the sequence number field that contains the value x , which specifies the initial sequence number of A.
2. B sends a SYN and ACK segment to A, where the sequence number field contains the value y , which specifies the initial sequence number of B and the ack field contains the received value x increased by one.
3. A sends an ACK segment to B, where the sequence number field contains the value $x + 1$ and the ack field contains the received value y increased by one.

Packets that flow in the network can be captured using a network traffic dump tool, like dumpcap, wireshark or tcpdump. All of them can visualize and store the captured data, while the last two can also analyze the captured packets. In wireshark data from a network interface are "dissected" in frames, segments and packets, understanding where they begin and end. Then, they are interpreted and visualized in the context of the recognized protocol. When we captures packets, we collect a lot of them, so to make our life easier we can use filters. In fact, they allows us to only focus on requested packets or certain activity by network devices. Filters are divided in : display filters in order to inspect only the packets we want to analyze once the data has been processed, and capture filters in order to limit the amount of network data that goes into processing and is getting saved. In the second one the packets not captured are lost, where in display filters we display only captured packets matching the filter, without discard them. Frames are collected from the interface and passed to several, consecutive, "dissectors" one for each layer (they are passed from the bottom layer to the upper one). Protocols can be detected in two ways :

- **Directly** : if a frame has the field that explicitly states which protocol it is encapsulating.
- **Indirectly** : with tables of protocol/port combinations and heuristics.

We can capture network traffic in different modes :

- **promiscuous mode** : cannot fetch every packet due to network segmentation.
- **physical tap** : device that mirrors traffic passing between two network nodes.
- **port mirroring** : a port of the switch which mirror all the traffic.
- **aggressive approaches** like :
 - **ARP poisoning** : send ARP replies to steal IP addresses
 - **MAC flooding** : fill the CAM table to make switch act like an hub.
 - **DHCP redirection** : exhausts IP addresses of the pool and then pretends to be the default gateway of the network with the new DHCP requests.
 - **ICMP redirection** : use ICMP type 5 message to indicate a better route.

Now, the question is How to prevent packet capture ? We can use two different approaches :

- **Dynamic address inspection** : validates ARP packets with IP-to-MAC inspection by intercepting every ARP request and dropping the invalid ones (placed on switches).
- **DHCP snooping** : always 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.

3 IPv6 introduction

IPv4 provides for a maximum 4.29 billion 32-bit addresses, which seemed like more than enough addresses when IPv4 became a standard in 1980. The number of IP addresses needed today far exceeds the world's population for several reasons. First of all, IPv4 addresses are often allocated in groups of addresses, as network addresses. Places such as companies, schools and airports are allocated network addresses for their users. But a much larger reason we need so many more IP addresses is the number of devices per

person that are being connected to the Internet. IPv6 provides more addresses than IPv4. In comparison, the 128 bit IPv6 address space provides for 340 undecillion addresses (i.e. approximately 2^{128}). The number of people accessing the Internet is increasing dramatically. Even with short-term solutions like NAT, we are in the final stages of public IPv4 address availability. There has also been extraordinary growth in the number of devices connected to the Internet. IANA assigns IPv4 addresses to the five Regional Internet Registries (RIRs) in /8 address blocks. Using this address space, RIRs then redistribute the IPv4 network addresses to ISPs and other end customers. On January 31, 2011, IANA allocated the last two blocks of IPv4 address space to RIRs. At this point, IANA had now run out of IPv4 addresses. This doesn't mean that IPv4 addresses are no longer available for end customers, but they can still get them from most ISPs. However, many ISPs are severely limited. So, in the early 1990s, IETF began the development of a new version of IP known as **IP Next Generation**, which later became **IPv6**. However, IPv6 was a long-term solution, and a short-term solution was needed immediately. Several short-term solutions were put into place. Two of the most important were **CIDR** and **NAT** with private IPv4 addresses. We know how CIDR allocates IP addresses. NAT along with private IPv4 addresses has been the reason IPv4 has survived all these years. It allows multiple hosts using private IPv4 addresses within their internal network to share one or more public IPv4 addresses when accessing the global Internet. NAT has also some critical points. At the very least, NAT means that our routers, application gateway and other devices must perform extra processing to make NAT work, which also causes latency. Other points are that it breaks peer-to-peer networking and accessing our "hidden" system from other networks. The IETF never intended NAT to be used for security. We want to say that in the late 1970s, a family of experimental protocols, known as **Internet Stream Protocol (ST)**, and later **ST2**, was developed. ST was an experimental resource reservation protocol intended to provide Quality of Service (QoS) for real-time multimedia applications such as video and voice. Although it was never recognized as **IPv5**, when encapsulated in IPv4, ST uses IP version 5.

3.1 Notation and types

IPv6 is much more than just a larger source and destination IP address. Developers of IPv6 took this opportunity to not only improve IP but also many of the protocols and processes related to IP. In fact, the benefits on IPv6 include : larger address space, stateless autoconfiguration, end-to-end

reachability without private addresses and NAT, better mobility support, peer-to-peer networking easier to create and maintain, and services like VoIP and QoS become more robust. An IPv6 address is 128 bits in length and hexadecimal is the ideal number system for representing long strings of bits. Every 4 bits can be represented by a single hexadecimal digit, for a total of 32 hexadecimal values from 0000 to FFFF. As described in RFC 4291, the preferred form to represent an IPv6 address is `x:x:x:x:x:x:x`. Each `x` is a 16-bit section that can be represented using up to four hexadecimal digits, with the sections (also called **hextets**) separated by colons. The result is eight 16-bit sections for a total of 128 bits in the address. There exists some rules in order to reduce the notation involved in the preferred format such as :

1. **omit leading 0s** : one way to shorten IPv6 addresses is to omit leading 0s in any hextet. This rule applies only to leading 0s and not to trailing 0s (because omitting both would cause the address to be ambiguous).
2. **omit all 0s hextets** : the second rule for shortening IPv6 addresses is that you can use a double colon (`::`) to represent any single, continuous string of two or more hextets consisting of all 0s. If there are multiple possible reductions, RFC 5952 states that the longest string of zeroes must be replaced with double colon and if they are equal then only the first string of 0s should use the `::` representation.

An IPv6 address can be **unicast**, **multicast** and **anycast**. A unicast address uniquely identifies an interface on an IPv6 device. A packet sent to a unicast address is received by the interface that is assigned to that address. Similar to IPv4, a source IPv6 address must be a unicast address. In the following sections we will deeply describe the previous types of IPv6 addresses.

3.1.1 Unicast addresses

An unicast IPv6 address in turn can be :

- **Global Unicast Address** : it's a globally routable and reachable address in the IPv6 Internet. They are equivalent to public IPv4 addresses. Its generic structure has three fields :
 - **Global Routing Prefix** : it's the prefix or network portion of the address assigned by the provider to the customer site.

- **Subnet ID** : the subnet id is a separate field for allocating subnets with the customer site. Unlike with IPv4, it's not necessary to borrow bits from the Interface id (host portion) to create subnets. The number of bits in the subnet id falls between where the GRP ends and where the interface id begins.
- **Interface ID** : it identifies the interface on the subnet, equivalent to the host portion of an IPv4 address. The interface id in most cases is 64 bits.

Except under very specific circumstances, all end users will have a global unicast address (or more than one). It's range is from 2000::/64 to 3fff:fff:fff:fff::/64.

- **Link-Local Unicast Address** : a link-local address is an unicast address that is confined to a single link, a single subnet. They need to be unique on the link and don't need to be unique beyond the link. Therefore, routers do not forward packets with a link-local address. Link-Local Unicast addresses are in the range of fe80::/10 to febf::/10. The following 54 bits are recommended to be all 0s and 64 bits dedicated to the Interface id. They are created in two ways :
 - **Automatically** : an IPv6-enabled device must have a link-local address. By default, devices automatically create their own link-local unicast addresses. The prefix is typically fe80::/64, followed by a 64 bit interface id that is automatically generated in one of the two ways :
 - * **EUI-64 generated** : the IEEE defined the EUI process using the interface's Ethernet MAC address to generate a 64-bit interface id. When EUI-64 is used to create a link-local address, the fe80::/64 prefix is prepended to EUI-64 generate interface id. An Ethernet MAC address is 48 bits, a combination of a 24 bit Organizationally Unique identifier (OUI) and a 24 bit Device Identifier, written in hexadecimal. The EUI process is an insertion of 16-bit value of fffe between the 24 bit OUI and the 24 bit device identifier, with the Universal/Local bit flipped (7th bit of the first byte).
 - * **Randomly generated** : EUI-64 is a convenient technique for automatically creating a 64 bit interface id from a 48 bit MAC address. However, it introduces a concern for some users : the ability to trace an IPv6 address to the actual

physical device using the 48 bit MAC address. To alleviate this privacy concern, devices can use randomly generated 64 bit interface ids.

- **Statically** : the disadvantage of an automatically generated link-local address is that the long interface ID is difficult to recognize. It's much easier to use a simpler, manually configured link-local address that is easier to identify.

3.1.2 Multicast addresses

Multicast is a technique in which a device sends a single packet to multiple destinations simultaneously (one-to-many), in contrast to a unicast address, which sends a single packet to a single destination (one-to-one). An IPv6 multicast address has the prefix `ff00::/8`, which defines a group of devices known as a multicast group. It's the IPv4 equivalent of `224.0.0.0/4`. A packet sent to a multicast group always has a unicast source address. A multicast address can only be a destination address and can never be a source address. The typical structure of a multicast address is the following :

- `ff00::/8` : the first 8 bits are 1-bits(ff), reserved for IPv6 multicast.
- **Flags** : the next 4 bits are allocated for flags. The fourth flag is the transient flag, which denotes two types of multicast addresses :
 - **Permanent (0)** : these addresses are assigned by IANA and include both well-known and solicited-node multicast.
 - **Non permanent (1)** : these are transient or dynamically assigned, multicast addresses. They are assigned by multicast applications.
- **Scope** : the scope field defines the range to which routers can forward the multicast packet (0 reserved, 1 interface-local scope, 2 link-local scope, etc.).
- **Group id** : the next 112 bits represent the group id.

There are two types of multicast addresses :

- **Assigned** : RFC 2375 defines the initial assignment of IPv6 multicast addresses that have permanently assigned Global IDs. IANA maintains the list of well-known IPv6 multicast addresses in a registry called

IPv6 Multicast Address Space Registry. For example when the group id value is one, then the multicast packet is sent to all network devices, when is 2 to all routers in the network, 5 to all OSPF router, and so on.

- **Solicited-Node** : a solicited-node multicast address is automatically created and assigned to an interface for every global unicast address, unique local address, and link-local address on that interface. These multicast addresses are automatically generated using a special mapping of the device's unicast address with the solicited-node multicast prefix `ff02:0:0:0:0:1:ff00::/104`.

One of the benefits of using a layer 3 multicast address is that it is mapped to a layer 2 Ethernet MAC address. This allows the frame to be filtered by the switch. This means these packets will only be forwarded out ports where there are devices that are members of the multicast group.

3.2 Addressing

Using **ICMPv6 Neighbor Discovery Protocol** Router Solicitation and Router Advertisement messages, hosts determine how to obtain their IPv6 addressing information dynamically. If SLAAC is used, the host can automatically obtain IPv6 addressing information, such as prefix, prefix length, and a default gateway address. The default gateway address is a link-local address, the IPv6 source address of the Router Advertisement message. ICMPv6 ND includes similar processes as in IPv4, such as address resolution, router discovery, and redirect, but also with some significant differences like prefix discovery, Duplicate Address Detection (DAD) and Neighbor Unreachability Detection (NUD). Neighbor Discovery uses five ICMPv6 messages :

- **Router Solicitation** message —
- **Router Advertisement** message — for router-device messages used with dynamic address allocation
- **Neighbor Solicitation** message —
- **Neighbor Advertisement** message — for device-device messages used for address resolution
- **Redirect** message — for router-device messages used for better first-hop selection.

A host sends a Router Solicitation message when it needs to know how to dynamically obtain its addressing information. This typically occurs during startup and is the default on most host operating systems. A Cisco router sends Router Advertisement messages every 200 seconds by default, and it also sends a RA message upon receiving a Router Solicitation message from a device. The RA message is a suggestion to devices on the link about how to obtain their addressing information dynamically. The RA message is sent to all IPv6 devices with a multicast address. The RA message contains three flag :

- **Address Autoconfiguration** flag (A flag) : when set to 1, which is the default setting, this flag tells the receiving host how to use SLAAC to create its global unicast address. SLAAC allows the host to create its own GUA by combining the prefix in the RA message with a self-generated interface id (using EUI or randomly generated).
- **SLAAC with Stateless DHCPv6** flag (O flag) : when set to 1 (default setting is 0), this flag tells the host to obtain other addressing information, other than its GUA, from a stateless DHCPv6 server. This information may include DNS servers addresses and a domain names.
- **Managed Address Configuration** flag (M flag) : when set to 1 (default setting is 0), this flag tells a host how to use a stateful DHCPv6 server for its GUA and all other addressing information. This is similar to DHCP for IPv4. The only information the host uses from the RA message is the RA's source IPv6 address, which it's used as the default gateway address.

In the second case after a device generates one or more addresses using SLAAC, it contact a stateless DHCPv6 server for additional information. Remember that a stateless DHCPv6 server doesn't allocate or maintain any IPv6 global unicast addressing information. A stateless server only provides common network information that is available to all devices on the network, such as a list of DNS server addresses or a domain name. In particular, a host after obtaining its own GUA sends out a DHCPv6 SOLICIT message to all DHCPv6 servers multicast addresses. One or more DHCPv6 server respond with a DHCPv6 ADVERTISE message, indicating that they are available for DHCPv6 service. The host responds to the selected server by sending an information REQUEST message, asking for other configuration information. The selected DHCPv6 server responds with a REPLY message

that contains the other configuration information.

In the third case, stateful DHCPv6 doesn't use SLAAC to generate a GUA. A stateful DHCPv6 server provides IPv6 GUAs to clients and keeps track of which devices have been allocated which IPv6 addresses. A significant difference between stateful DHCPv6 and DHCPv4 is the advertising of the default gateway address. In IPv4, the DHCPv4 server usually provides the default gateway. In IPv6, only the router transmitting the ICMPv6 Router Advertisement can provide the address of the default gateway dynamically. There is no option within DHCPv6 to provide a default gateway address. A host after receiving the RA uses the source IPv6 address of the RA as its default gateway address. Then addressing and other configuration information is available from a stateful DHCPv6 server. So, the host sends out a DHCPv6 SOLICIT message to all DHCPv6 servers multicast addresses, searching for a DHCPv6 service. One or more DHCPv6 server respond with a DHCPv6 ADVERTISE message, indicating that they are available for DHCPv6 service. The host selects one DHCPv6 server by sending to it a DHCPv6 REQUEST message asking for addressing and other configuration information. The selected DHCPv6 server responds with a reply message that contains a GUA and other configuration information. Finally, the host performs DAD on the address received from the stateful DHCPv6 server to ensure that this address is unique.

To ensure unicast address uniqueness, address resolution procedure includes **Duplicate Address Detection (DAD)**. The device sends a Neighbor Solicitation message for its own IPv6 address to detect whether another device on the link is using the same address. If a Neighbor Advertisement message is not received, the device knows its address is unique on the link.

3.2.1 Prefix Delegation process

In the world of IPv4, most internal networks use a private IPv4 address space for internal devices and NAT at the edge router to translate an address to a globally routable public IPv4 address. This is a common mechanism for most home networks and is partly responsible for keeping IPv4 alive for so many years. Avoiding the complications and problems with address translation, IPv6 uses a different technique. One of the methods IPv6 uses is DHCPv6 with the **Prefix Delegation** option, which provides a mechanism for automated delegation of globally routable IPv6 prefix from a provider's router to a customer's router using DHCPv6. In this process are involved

two routers :

- **Requesting Router (RR)** : this is the router that acts as the DHCPv6 client, requesting the prefix(es) to be assigned.
- **Delegating Router (DR)** : this is the router that acts as the DHCPv6 server, responding to the requesting router's IPV6 prefix request.

Lets describe in details the Prefix Delegation process. First the RR's ISP facing interface needs an IPv6 address, that can be dynamically obtained using SLAAC, stateless or stateful DHCPv6 server. Then the RR initiates DHCPv6-PD in its SOLICIT message by including a request for an IPv6 prefix. The REPLY message from the DR (the ISP router) includes the IPv6 prefix. This is the prefix that the RR can use for its own internal network (i.e. a prefix that the RR can use to allocate addresses to its clients).

3.3 Comparing the IPv4 and IPv6 headers

In comparing the two headers, notice that the IPv6 header takes advantage of 64-bit CPUs, respect to the 32-bit CPUs of the IPv4 header. As a result, all IPv6 fields start on an even 64-bit boundary or a multiple of 64. The advantage is that 64-bit CPUs can read one 64-bit-wide memory word at time. However, this structure doesn't negatively affect 32-bit CPUs because a 64-bit boundary is also a 32-bit boundary. A quick comparison of the two headers reveals that the IPv6 header is a simpler protocol with fewer fields than its IPv4 counterpart. This makes IPv6 a leaner protocol and provides more efficient processing. Both IPv4 and IPv6 begin with the **Version** field, which contains the version number of the IP header. The IPv4 **Internet Header Length (IHL)** field, is the length of the IPv4 header in 32-bit words, including any optional fields or padding (i.e. IPv4 has a variable header length). IPv6 doesn't have an IHL field because the main IPv6 header has a fixed length of 40 bytes, which allows for more efficient processing. The IPv4 **Type of Service (ToS)** field and the IPv6 **Traffic class** field are identical fields. In particular, they are used to specify what type of treatment the packets should receive from routers. This information helps to provide QoS features by offering different degrees of precedence. The IPv6 **Flow Label** field, is a new field used to tag a sequence of flow of IPv6 packets sent from a source to one or more destination nodes. This field can be used by a source to label sequences of packets for which it requests special handling by the IPv6 routers, such as "real-time" service. The IPv4 **Total Length** field is the length of the entire IPv4 packet, measured in

bytes, including the IPv4 header and the data. The IPv6 **Payload Length** field is a 16-bit field that indicates the length in bytes of just the payload following the main IPv6 header. It does not include the main IPv6 header, but only the payload length and extension headers length. The design of IPv4 accommodates MTU differences by allowing routers to fragment IPv4 packets when an MTU along the path is smaller than the sender's MTU. If a router receives an IPv4 packet that is larger than the MTU of the outgoing interface, this packet can be fragmented, depending on the options in the IPv4 header. Sometimes packets are fragmented into multiple packets at the source. The final destination of the IPv4 packet is responsible for reassembling the fragments into the original full-size IPv4 packet. Fragmentation divides IPv4 packets so that they can be forwarded out a link that doesn't support the size of the original packet. IPv4 **Identification, Flags and Fragment Offset** fields are used for packet fragmentation and reassembly. Unlike in IPv4, an IPv6 router doesn't fragment a packet unless it's the source of the packet. Intermediate nodes do not perform fragmentation. When an IPv6 router receives a packet larger than the MTU of the output interface, the router drops the packet and sends an ICMPv6 Packet Too Big message back to the source. This message includes the MTU size of the link in bytes so that the source can change the size of the packet for retransmission. A device can use **Path MTU Discovery** to determine the minimum link MTU along the path. The IPv4 **Protocol** field indicates the protocol carried in the data portion of the IPv4 packet. IPv6 has a similar field, the **Next Header** field, that specifies the type of header expected after the main IPv6 header. The same values used in the IPv4 Protocol field are used in the IPv6 Next Header field, along with some additional values for IPv6. Some of the most common values for both protocols are 6 for TCP and 17 for UDP. The IPv4 **Time To Live (TTL)** and IPv6 **Hop Limit** fields ensure that packets do not transit between networks for an indefinite period of time, as in the case of a routing loop. These fields are decremented by one each time a router receives the IPv4 or IPv6 packet. When the field contains the value zero, the packet is discarded, an ICMPv4 or ICMPv6 Time Exceeded is sent to the source of the packet. A **checksum** field for the IPv4 header is provided for protection against any corruption in transit. Each router along the path verifies and recomputes this field. If the checksum fails, the router discards the packet. There is no checksum field in the IPv6 header. The designers of IPv6 didn't include this field because layer 2 data link technologies such as Ethernet perform their own checksum and error control. Because there is no checksum field in IPv6, the UDP checksum is mandatory in IPv6. The most notable and recognizable difference in IPv6 compared to IPv4 is

the expansion of the source and destination address from a 32-bit address in IPv4 to a 128-bit address in IPv6. The IPv4 **Option** field is optional. It's variable in size and rarely used, so it isn't usually included in most IPv4 packets. The IPv4 **Padding** field is used only if the IPv4 Options field is used and the size of the IPv4 header is no longer a multiple of 32 bits. When this is the case, the IPv4 header is padded with bits that are 0s so that it ends on a 32-bit boundary. Extension headers are an important addition to IPv6. They are optional headers that add flexibility and allow for future enhancements to IPv6. The main IPv6 header include a Next Header field, which has one of two purposes : to identify the protocol carried in the data portion of the packet or to identify the presence of an extension header. The next header field indicates an additional header known as extension header. Immediately following the mandatory main IPv6 header, there can be zero, one or several extension header. It also allow the main IPv6 header to be a fixed size for more efficient processing. IPv4 uses options and padding fields to accomplish what the extension headers do in IPv6. The next header field is used to chain together multiple IPv6 headers and the data portion of the packet at the end of the chain. RFC 2460 recommends that when multiple extension headers are used in the same packet, those headers should appear in the following order :

1. Main IPv6 header
2. Hop-by-Hop options header
3. Destination option header
4. Routing option header
5. Fragment header
6. Authentication Header (AH)
7. Encapsulating Security Payload (ESP) header
8. Destination Options header
9. Upper-layer protocol header.

3.4 Link-local attacks

In this section we want to explain in details some common types of attacks over link-local addresses.

Network eavesdropping There is somebody within the network that capture packets transmitted by others and read the data content in search of sensitive information (such as password and session tokens). A possible solution to avoid data eavesdropping is to use encryption. In fact, using a strong encryption algorithm, the attacker will be not able to decrypt sensitive information. It's important not only to protect the content of the packets, but also some information that are used to move the packets (in this way we enforce privacy measure), using for example **TOR** or a **VPN**. Network eavesdropping is done by using tools called **network sniffers**, which goal is to focus on really breaking the confidentiality of the data. In particular, they analyze the collected data, filters all the valuable information and provide them at user level. They don't generate any kind of traffic, i.e. work in passive mode. To successful perform an eavesdropping attack, you need to be along the path of the communication. To realize this type of attack, first of all your network interface should be placed in promiscuous mode, i.e. you need to remove from the drivers of your network card the check of your MAC address, in order to accept every network packets. In wireless LAN the things are even worst, in fact you can potentially receive not only the packet of your network but also the packets of other networks that have a station that is in the range of your wireless card. So, we can place a sniffer in several places like :

- **Non-switched LAN (LAN with HUBs)** : it's the ideal case, because the hub duplicates every frame to all the ports.
- **LAN with switches** : in the switch the whole network is segmented, so we try to alter some of the mechanism of the switch, trying to make it to send the packets also in our segment (see MAC flooding). Or we can also use an ARP spoof attack, convincing the source of the packet that to send the packet to its intended destination, must send the packet to us. Lets see in details these two attacks :
 - **MAC flooding** : before considering the switches there were the bridges. The bridges were the first devices that were able to connect two network segments. The idea is to extends the broadcast domain but also reducing the collisions. These devices uses the **store & forward** technique, with which read and regenerate a frame only if needed. If we imagine that the bridge can have multiple port (i.e. a switch), then the mechanism is exactly the same, analyze the packet and understand on which port is connect a segment where the desired destination is connected. Whether

we generate a packet we insert our MAC address in the packet as source MAC address and the switch builds this association between a port, a segment and a list of the hosts in that segment considering the MAC source address. The MAC address is saved in the **CAM Table**, using the content itself to understand where to save that information in memory. What happens when a switch receive a packet in case of destination MAC unknown ? It's clear that it doesn't know on which port is the segment that host the destination MAC, so it replicates the packets on all the ports (flooding on the network). This behavior is used for the **CAM overflow** attack, where the CAM table is filled with useless MAC addresses and then whenever we have no more space to store a MAC address, a cache miss, we have to replicate the packets. Usually switches use hash to place MAC in CAM table, i.e. like hashed lists, where buckets can keep a limited number of values; if the value is the same there are n buckets to place CAM entries, if all n are filled the packet is flooded. In order to avoid this type of attack, we can employ the port security in switches :

- * allows us to specify MAC addresses for each port, or to learn a certain number of MAC addresses per port.
 - * upon detection of an invalid MAC the switch can be configured to block only the offending MAC or just shut down the port.
- **ARP spoofing** : An ARP request message should be placed in a frame and broadcast to all computer on the network. Each computer receives the request and examines the IP address. The computer mentioned in the request sends a response, while all other computers process and discard the request without sending any response. We need ARP to resolve an IP address to MAC address. For example, host A wants to communicate with host B. Host A realize that host B is in the same network, so they can communicate directly using Ethernet. So, we need a MAC address of B but A don't know it. Host A sends an ARP request, including the IP address that he's looking for, to all network hosts. Subsequently, host B recognize its own IP address and will answer sending back an ARP-reply to host A. The ARP reply has the MAC address of B as source and MAC address of A as destination, picked from the original ARP request. Then

host A can finally send directly the IP packets to host B. The main problem with this protocol is that it's untrusted, i.e. we cannot really trust the information that you see in ARP request and ARP reply. Whenever we receive an ARP reply, we store this information in an **ARP table**, which is a temporary table that is filled with this IP-MAC pairings. Whether we want to send a packet to a host in the same network, we check if the ARP table contains the MAC address corresponding to the target IP address. In a positive case, we use exactly that MAC address, otherwise we proceed as before (ARP request - ARP reply). Another mechanism used by this table is the **address aging**, which removes unused MAC addresses after a specific timeout. Whenever we receive an ARP reply from the network, we extract the IP-MAC pairing and update the corresponding entry in the ARP table. This activity is typically performed with packets that are called **Gratuitous ARP**. It's an ARP reply that is not really an answer to an ARP request, but it's something performed by ourselves, used to announce our association between IP and MAC address. In this way, we avoid duplicate IP addresses in the network. What if we misuse the Gratuitous ARP ? A situation in which a host frequently announces its pairing using not his IP address but another one, for example the address of the network default gateway. So, all the other network hosts will be fooled to think that the default gateway is no more the MAC of the real router, but now the new MAC is the one of the attacker. In this case, the attacker will receive all the packets that are intended to go outside the network. What we have is an hijacking in the local network, i.e. you can pretend to be anybody in your network. The very first type of attack is the **Denial of Service**. Another type is the **Man In The Middle** attack, in which the attacker intercepts the traffic and reroutes it to get the reply, then forwards the reply back.

- **Wireless LAN** : if we used weak encryption or no encryption, the scenario becomes equivalent to LAN with HUBs. If we use more recent encryption tools like WPA then we are protected, i.e. all the transmissions that we can potentially capture, are encrypted.

IPv6 Neighbor Discovery threats ICMPv6 Redirect can be abused in some way to create a rogue router. Whenever we have a host that sends a

packet to the router because it has to be sent to a network X and the router reaches another router in the same network, then the first router forward the traffic to the second one, but sends also an ICMPv6 Redirect message to the host, informing he that he can reach network X with one hop less. In IPv6 a router can know if two hosts are not really in the same network, but if they are in the same link local. Some types of attacks based on IPv6 ND are :

- Neighbor Solicitation / Advertisement spoofing : same as ARP spoofing, i.e. pretends to be somebody else.
- DAD DoS attack : imagine you use SLAAC in order to get a Global Unicast address, and the attacker replies to you that the address is used. This scenario is repeated and repeated, generating a DoS attack. At some point the host stop to send Neighbor Solicitation, deciding to not be part of the network.
- spoofed Redirect message, default router compromise, malicious last hop router (router threat).

Rogue Router Advertisement What if the RA comes from a malicious router ? One of the most common problem due to rogue RA is the **VPN bypass**. Whenever you connect to a VPN, you will be part of a new network. Usually, this network is an IPv4 network and you're supposed to use that network for all the communication that should go in the VPN. If you're able to inject a rogue RA, you can fool the host and make it to use your own network, the one injected in the RA, instead of the VPN network. If the attacker is able to inject a new DNS or a new default gateway, he can sniff all client traffic, attempt MITM attacks, impersonate server/systems and capture presented user credentials, gain access into the other networks of the systems. The countermeasures are at network level. It means that your switches/routers should protect you against these types of attacks. The mechanism is not called anymore port security but is called **RA guard**. We can write some policies that states from which port the RAs have to come, i.e. it's the port where the default gateway is supposed to be. If we receive a RA from another port, we just discard it.

Router Advertisement flooding We can also have a DoS attack, using the RA flooding. A host can potentially generate an IPv6 address for each prefix that he has received from RAs. The same reasoning is applied for the routing information.

DHCP rogue server It's a situation in which an attacker is able to request all the available DHCP addresses. Once the addresses are gone, an attacker could use a rogue DHCP server to provide addresses to clients. Since DHCP responses include DNS servers and default gateway entries, the attacker can pretend to be anyone.

4 Internet firewall

Firewall are a very effective type of network security. In building construction, a firewall is designed to keep a fire from spreading from one part of the building to another. In theory, an Internet firewall serves a similar purpose : it prevents the dangers of the Internet from spreading to your internal network. It serves multiple purposes :

- it restricts people to entering at a carefully controlled point.
- it prevents attackers from getting too close to your other defenses.
- it restricts people to leaving at a carefully controlled point.

A firewall is a device that besides acting as a router, also contains and implements rules to determine whether packets are allowed to travel from one network to another. We have several ways to enforce the protection mechanism of the network :

- regulate which traffic is allowed
- protect the traffic with encryption
- monitor the traffic for "bad behavior"
- monitor the hosts for "bad behavior".

The configuration will depend on the security policy to be fulfilled (see CIA targets). When we talk about firewall, we have to take some assumptions :

- we have to know our policy stating what is allowed and not allowed.
- we have to understand the good and bad traffic by its IP address, TCP port numbers, etc.
- the firewall itself is immune to penetration.

4.1 Security strategies

It's important to understand some of the basic strategies employed in building firewalls and in enforcing security at your network.

Least Privilege Basically, the principle of least privilege means that any object 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 particular attacks.

Defense in depth It doesn'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.

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 network and the Internet is such a choke point; anyone who's going to attack your network from the Internet is going to have to come through that channel, which should be defended against such attacks. A choke point is useless if there is an effective way for an attacker to go around it.

Weakest links Smart attackers are going to seek out the weak point of the security mechanism and concentrate their attentions there. We need to be aware of the weak points of our defense so that we can take steps to eliminate them, and so that we can carefully monitor those we can't eliminate. We should try to pay attention equally to all aspects of our security, so that there is no large difference in how insecure one thing is as compared to another.

Fail-safe stance If some network node fails, then it's 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. The biggest application of this principle in network security is in choosing your network's stance with respect to security. In general, there are two fundamental stances that we can take with respect to security decisions and policies : the **default deny stance** to specify what we allow and prohibit

everything else, and the **default permit stance** to specify what we prohibit and allow everything else.

Universal participation If someone can simply opt out of your security mechanisms, then an attacker may be able to attack you by first attacking that exempt person's system and then attacking your network from the inside. In other words, the idea is to be able to convince all the users to choose the security mechanism used by a particular system.

Diversity of defense It's closely related to depth of defense but takes matters a bit further; it's the idea that we need not only multiple layers of defense, but different kinds of defense. If we're not careful, we can create diversity of weakness instead of diversity of defense. If we have two different packet filters, one of them in front of the other, then using different products will help protect us from weaknesses in either one. If we have two different packet filter, each separately allowing traffic to come in, then using different products will merely make us vulnerable to two different sets of problems instead of one.

Simplicity Simplicity is a security strategy for two reasons. First, keeping things simple makes them easier to understand; if you don't understand something, you can't really know whether or not it's secure. Second, complexity provides nooks and crannies for all sorts of things to hide in; Complex programs have more bugs, any of which may be a security problem.

4.2 Packet filters

The very first packet filter we want to describe is the **host based** one. It specifies the packets that can be received and sent (examples are the windows firewall and Linux iptables). The type of router used in a packet filtering firewall is known as a **screening router**. A screening router looks at packets more closely respect to how an ordinary router does. In addition to determining whether or not a packet can be forwarded to its destination, a screening router also determines whether or not it should. "Should" or "should not" are determined by the network security policy, which the screening router has been configured to enforce. On the other hand, it's very flexible; in fact we can permit or deny protocols by port number, but it's hard to allow some operations while denying others in the same protocol, or to be sure that what's coming in on a given port is actually the protocol we wanted to allow. In addition, it gives us no depth of defense. If the

router is compromised, we have no further security. A screening router is an appropriate firewall for a situation where :

- the network being protected already has a high level of host security.
- the number of protocols being used is limited, and the protocols themselves are straightforward.
- we require maximum performance and redundancy.

Screening routers are most useful for internal firewalls and for networks that are dedicated to providing services to the Internet. The most common way to have protection for our network is to use **dual-homed host** architecture. It's built around the dual-homed host computer, a computer that has at least two network interfaces. Such a host could act as a router between the networks these interfaces are attached to; it's capable of routing IP packets from one network to another. However, to use a dual-homed host as a firewall, we disable this routing function. Systems inside the firewall can communicate with the dual-homed host, and also external system can communicate with it, but they can't directly communicate with each other. Dual-homed hosts can provide a very high level of control. If we aren't allowing packets to go between external and internal networks at all, then we can be sure that any packets on the internal network that has an external source is evidence of some kind of security problem. In contrary, they are not high-performance devices. In fact, a dual-homed host has more work to do for each connection than a packet filter does, and thus needs more resources. An attacker who can compromise the dual-homed host has full access to our network, so we must guarantee an impeccable host security. A dual-homed host is an appropriate firewall for a situation where :

- traffic to the Internet is small.
- traffic to the Internet is not business-critical.
- no services are being provided to Internet-based users.
- the network being protected doesn't contain extremely vulnerable data.

Another well known architecture is the **screened host**. Whereas a dual-homed host architecture provides services from a host that is attached to multiple networks, a screened host architecture provides services from a host that is attached to only the internal network, using a separate router. In this architecture the primary security is provided by packet filtering. This

host is called **bastion host**, which is a hardened computer used to deal with all traffic coming to a protected network from outside. What does it means hardening ? Hardening is the task of reducing or removing vulnerabilities in a computer system :

- shutting down unused or dangerous services. Strengthening access controls on vital files.
- removing unnecessary account permissions.
- using "stricter" configurations for vulnerable components, such as DNS, FTP, etc.

There are some disadvantages to the screened host architecture. The major one is that if an attacker manages to break in to the bastion host, nothing is left in the way of network security between the bastion host and the rest of the internal hosts. A screened host architecture is appropriate when :

- few connections are coming from the Internet.
- the network being protected has a relatively high level of host security.

Another architecture that we want to describe is the **screened subnet**. It adds an extra layer of security to the screened host architecture by adding a **perimeter network** that further isolates the internal network from the Internet. Why do this ? By their nature, bastion hosts are the most vulnerable machines on our network. Despite our best efforts to protect them, they are the machines most likely to be attacked because they are the machines that can be attacked. If, as in a screened host architecture, our internal network is wide open to attack from our bastion host, then this latter is a very tempting target. No other defenses are between it and our internal hosts. If someone successfully breaks into the bastion host in a screened host architecture, that intruder has hit the jackpot. So, by isolating the bastion host on a perimeter network, we can reduce the impact of a break-in on the bastion host. It gives an intruder some access but not all. The simplest type of screened subnet architecture, has two screening routers, each connected to the perimeter network. One sits between the perimeter network and the internal network (also known as **interior router**), and the other sits between the perimeter network and the external network (also known as **exterior router**). An attacker to break into the internal network with this type of architecture, must have to bypass both routers. Even if the attacker somehow broke into the bastion host, he would still have to get passed the

interior router. There is no single vulnerable point that will compromise the internal network.

A **demilitarized zone (DMZ)** is a special small network that is supposed to be protected from the outside, but it supposed to expose some services that a company intends to provide outside. DMZ is very often represented by means of firewall using the bastion host and the screening router. The first hop of the packets must be the bastion host and then it decides if they can be forwarded or not. We can segment the network in many different cases, for examples using a firewall to segment the network in several DMZs (also called VLAN).

4.3 Stateless firewall

In stateless firewall the packets are considered one by one. We receive a packet from one end of the filter, we look into it and decide if it can be forwarded or not. We can also take different decisions according to where the packet coming from (input or output interface). This is useful because we can protect the network from spoofed ip addresses. The packet filter operates at Data link, Network and Transport layer, observing source and destination addresses, and port numbers. So, image that we have :

1. our policy
2. we have expressed this policy in a formal language
3. and rewrote our policy in terms of the firewall syntax.

The general mechanism is constituted by a set of rules to be checked from top to bottom. The first matching rule is applied and one implicit rule (also called the default rule) is assumed if no rule matched. Usually, the safest default decision is to block everything. In particular scenarios, is not enough to consider only a set of rules, because we can't control the direction of the traffic. For this reason, the packet filter system to consider the direction of the packets usually uses the TCP flags (for example the ACK flag).

4.4 RAW, MANGLE, NAT and FILTER tables

So far we have focused our attention on the FILTER table, but it's useful to notice that there are other tables in the iptables world such as :

- **RAW** : it's used for altering connection tracking, but by default it's not loaded since it's used for specific reasons.

- **MANGLE** : it means to manipulate the packets, for example to change the TCP header, ToS, TTL, etc. It shouldn't be used for FILTERING or NAT.
- **NAT** : it's used for network address translation, but we will see it in details in the next section. In particular, it handles the following special targets : DNAT, SNAT, MASQUERADE and REDIRECT. The DNAT is used in the PREROUTING chain to transform the destination IP of incoming packets, SNAT is used in the POSTROUTING chain to transform the source IP of outgoing packets, MASQUERADE is similar to SNAT, but it will change the outgoing packets inserting the IP address of the interface that in that particular moment the firewall has.
- **FILTER** : in the previous section we have talked about stateless firewall, specifying that in some situation additional information are required in order to understand the traffic direction (egress filtering). Again, this is not enough, because somebody outside from our network can spoof some internal IP addresses. For this reason, we need to define rules based on from where packets are arriving (ingress filtering). Furthermore, we have some problems here : we take into account only a small number of parameters, we can't inspect the payload of TCP packets and there are some limitations about logging ability, no authentication facilities, and so on.

There exist a priority between the previous tables : in particular, if we represent with $A > B$ the meaning that A has higher priority with respect to B , then we have $RAW > MANGLE > NAT > FILTER$. We can also specify additional chains in the different tables, in order to perform additional checks on the packets. It's very similar to the standard chain mechanism, but we can jump to a different chain within the same table.

4.5 Stateful firewall

A stateful firewall with respect to its stateless counterpart, offers state tracking of established connections. It is also called dynamic packet filtering because the behavior of the system changes depending on the traffic it sees. State tracking provides the ability to do things that we can't do otherwise, but it also adds complications. First, the router has to keep track of the state, increasing the load of traffic they should manage. This opens them to a number of DoS attacks, and means that if the router reboots, packets may

be denied when they should have been accepted. Second, the router has to keep track of state without any guarantee that there's ever going to be a response packet (i.e. not all UDP packets have responses). For instance, DNS replies are supposed to arrive within 5 seconds, but reply times for name service queries across the Internet can be as high as 15 seconds; implementing the protocol specification (i.e. set of guidelines to avoid routers overload) will almost always deny a response that you wanted to accept.

4.6 Other types of firewalls

In this section we want to briefly describe some other types of firewall we could facing along the way.

Application-level proxy An application-level proxy is one that knows about the particular application it is providing proxy services for; it understands and interprets the commands in the application protocol.

Circuit-level proxy A circuit-level proxy is one that creates a circuit between the client and the server without interpreting the application protocol. The advantage of a circuit-level proxy is that it provides service for a wide variety of different protocols. Most circuit-level proxy servers are also generic proxy servers; they can be adapted to server almost any protocol. In contrary, not any protocol can be easily handled by a circuit-level proxy, because some of them could require some application-level knowledge (like FTP protocol). The disadvantage of a circuit-level proxy server is that it provides very little control over what happens through the proxy. Like a packet filter, it controls connections on the basis of their source and destination and can't easily determine whether the commands going through it are safe or even in the expected protocol. In general, circuit-level proxies are functionally equivalent to packet filters.

The most extreme version of the first approach is an application like Sendmail, which implements a store-and-forward protocol. The most extreme version of the second approach is an application like plug-gw, which accepts all data that it receives and forwards it to another destination. When we use the application gateway, for every incoming packet we verify the client application and then verify the user. While in the circuit level, for every incoming packet we consider the user authentication.

5 Network Address Translation

The internal network uses private IP addresses provided by IETF, and can change them for hosts/devices within this network without notifying the world outside. While IP addresses for hosts in the external network are unique and valid in this environment as well as in private networks, the addresses for hosts in the private network are unique only within this private network and may not be valid in the external network. The IP address binding in some cases may extend to transport level identifiers such as TCP/UDP ports. The address binding is done at the start of a session, and a traditional NAT device would allow hosts within a private network to transparently access hosts in the external/public network, in most cases. There are two variations to traditional NAT :

- **Basic NAT** : a block of public IP addresses is set aside for translating the addresses of hosts within a private domain as they originate sessions to the external domain. In fact, translation occurs for both outbound and inbound packets. For outbound packets from the private network, the source IP address and related fields such as IP, TCP, UDP and ICMP header checksums are translated. For inbound packets, the destination IP address and the checksums are translated.
- **Network Address Port Translation (NAPT)** : it also translates transport identifiers, e.g. TCP and UDP port numbers as well as ICMP query identifiers. This permits the transport identifiers of a number of private hosts to be multiplexed into the transport identifier of a single public IP address. The NAPT allows a set of hosts to share a single external address. For most of the SOHO routers, the private network usually relies on a single IP address, supplied by the ISP to connect to the Internet, and can change ISPs without changing the private IP addresses of the devices within the network, since these devices inside the network are not explicitly addressable by the external network. This latter point is also a security advantage.

The NAPT maps the private source address and source port number to a public source address and a public-side port number at the NAPT router for outgoing packets. Incoming packets, addressed to this public address and port pair, are translated to the corresponding local address and port. The NAPT translation table provides a one-to-one mapping entry between the two pairs : *private source IP address:port number* and the *public IP address:new port number*. With outgoing datagrams, the first pair is replaced

with the second one, and remote client will reply using this latter pair as the destination transport address. For incoming packets, the previous process is reversed. One of the more pressing problems encountered is that in which NATs commonly enforce an application model where a local, private, hidden host must initiate a transaction in order to create a hole in the NAT in order to allow the packets of the external host back into the private network. Some applications may wish to undertake a "referral", in which the correspondent host on the external side may want to pass the externally presented address and port details of the local host to a third party in order to start a further part of the transaction. By default, NAT routers block all incoming requests and only allow the response packets of outgoing requests to pass through it as a result of the available mapping entries. Many applications have had problems with NAT in the past in their handling of incoming requests. There are four major methods used for handling the connection to a server that exists between a private network and the outside network :

- Application Level Gateways (ALGs)
- Static port forwarding
- Universal Plug and Play (UPnP), Internet Gateway Device (IGD) protocol
- Traversal Using Relays around NAT (TURN).

In an IP packet we can translate source or destination address using two different methods :

- **Source NAT (SNAT)** : it means we want to change the IP address of the host that generates the packet with a public one. We can easily notice that source translation happens when a packet wants to go out from a private network. This is also called masquerade, because it acts like a mask applied to packet source address, that will be removed when the answer arrives. When the response packet arrives to the NAT router, we also apply a DNAT in order to translate the public destination address to a private destination address. In this section we are referring to SNAT, since first is applied a masquerade to the source address and then to the destination address. Typically, the considered router has a table called **NAT table** that uses to store several mappings between private and public addresses. We want to underline that the data stored in the NAT tables are temporary, i.e. when the connection will be closed, NAT table entries can be removed.

- **Destination NAT (DNAT)** : we need destination NAT when we want to have an internal host to receive a connection from the outside. We have a source client that wants to reach a server within our internal network. In other words, the server address isn't a private IP address but will be a public IP address that is the one assigned to the firewall external interface. When the firewall receives a packet from the outside that wants to reach our server, it has to translate the packet destination address into the server private IP address. This is also called **Port Forwarding** or **Virtual Server**. The idea is the following : whenever the firewall receives a connection request on the external interface, according to the requested port it will forward to a different host.

6 Virtual Private Networks

A VPN is 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 new network typically is based on the use of encryption deciding how and where to perform it and what parts of the communications should be encrypted. The VPN main goal is the usability, so that has to be in some way easy to use and to be exploited. This goal can be measured with the following metrics :

- **Transparency** : it means how much burden the users of the VPN should sustain to use it.
- **Flexibility** : how easily can be used the VPN and how many uses we can have of the VPN.
- **Simplicity** : how easy can be to deploy and use the VPN.

The VPN security goals are :

- **Confidentiality of data** : it means that we don't want that other guys access the data.
- **Integrity of data** : it means that any of the two endpoints can recognize that the transmission was done without some errors or changes.
- **Peer authentication** : we can know the authenticity of the data, i.e. the data that we receives is coming from the original sender.

We can also see some extended features about a VPN such as :

- **Replay protection** : we can distinguish between fresh and old data.
- **Access control** : we can limit the access to the network resources to the users that leverages the network.
- **Traffic analysis protection** : we can use some techniques in order to avoid some kind of analysis of encrypted data that can really reveal some details.

The **site-to-site** security means that we are using as an insecure channel Internet, to provide a virtual circuit between two endpoints that are two routers that trying to bind together two sites of the same institution. The idea is that all the traffic that is supposed to be routed between these two sites, instead of going as they are into the Internet, they are transformed and injected in this channel that they have established in Internet. We can also have a different type of security called **host-to-site** security, where one the two endpoints is a single host. The idea is that we again we are using Internet to provide the access to join a host to the network of one site, and the virtual circuit between the host and gateway provides and guarantee the security properties. Another type of security is the **host-to-host** security, where who is performing the encryption are the two endpoints in the network. We can also distinguish where we want to apply the security. In particular, referring to TCP/IP model we can implement the security at :

- **Physical layer** : it means that we are securing a wire. From the point of view of the flexibility is very bad, because is only a point-to-point security and thus hard to adapt to other contexts. From the point of view of the transparency, we are not changing anything from the point of view of the applications, because the encryption is only at physical level. A physical VPN is the less common to be used.
- **Data link layer** : we need to remember that the data link layer in some way supposed to carrying in the payload all the data of the application level and all the headers of the previous layers. In this case the confidentiality is on the link, i.e. all the data exchanged on this virtual cable are secure. Also in this case we have full transparency, so the users won't notice in any case that there is the VPN, but isn't flexible because it's again a n-to-point security.
- **Network layer** : it means includes some IP packets (not really IP packets because they're encrypted) in the packets header. It's very

useful, because we can keep the already deployed network and hide the contents of IP packets (this type of VPN is implemented with IPsec). It's very transparent, flexible and simple for site-to-site security. It's not so simple for host-to-site security, because from the point of view of the transparency we need some extra setup in the host.

- **Transport layer** : in this case the protection is end-to-end between processes. It's very good for site-to-site and host-to-site security. It's very good also for access control and transparency (this type of VPN is implemented with SSL/TLS).
- **Application layer** : it's not really a VPN, but we are making two applications to communicate in a secure way. This is the standard HTTPS realization. It's very good from the point of view of the transparency, protection of the data but it isn't flexible because we have to use a different VPN for each application we want to use. This is a common implementation used for host-to-host security.

6.1 SSL Tunneling

We've introduced the concept of tunneling, which means that we have established a tunnel between two endpoints and it provides some guarantees to all the data that pass through it such as encryption and authenticity. We can realize two types of security :

- **transport site-to-site** : we can what actually is the information that was given from one layer to the lower layer. In other words, we can image that the sender PDU is encapsulated in another PDU (IP packet) in order to allow it to cross the network that connects the two sites. In particular, the encapsulation takes place in the edge router on the source site, while the decapsulation in the edge router on the destination site.
- **tunneled site-to-site** : we can also consider site-to-site security with encryption, i.e. before encapsulating the PDU to be transported in another PDU, it's encrypted so that its contents cannot be seen or changed by hosts on the route. The encryption can take place in the edge router on the source site, whereas the decryption in the edge router on the destination site.

The tunnel is the way in how we realize a VPN. Lets start with an introduction with **Secure Socket Layer (SSL)**. SSL was originally designed

by Netscape to offer security for client-server session. There are many version of SSL, and starting from the version 3.0 became TLS standard with small changes. Clearly, the security is the Transport layer. Since it works at transport layer, we can have at the lower layer of the stack exactly the same protocols that we have in the standard Internet. Lets review shortly how HTTPS works (i.e. HTTP on top of TLS). When you connect to a website that provide HTTPS security, remember that the only thing that it does it's uses HTTP on top of TLS. This means that before using HTTP, instead of using the standard TCP, we use TLS to establish the secure channel. The first packet is the one intended to obtain the first part of information that represent the certificate of the server. Is important to have the server certificate because we want to be really sure that the key we will use to exchange the data is the one that belongs to the website. The certificate is a way to bind the key of the website to the domain name of the website. It has a public key, which is the one we have to use to encrypt the communication that has to go to the server. This binding is guaranteed by a digital signature of our Certificate Authority. We can use the server public key to verify the digital signature, and if this check is positive we are sure that this is the real public key of the server. Now, we can generate a symmetric key that is supposed to be used during the secure communication. We can encrypt this key with the asymmetric key of the website, and sent to insecure Internet to reach the website. Then the website can decrypt it using its own private key. Now, both parties are agrees on a secure key and it can be used to cipher a channel between them. Between HTTP and TCP we insert SSL protocol, also called **Record Protocol**. Before being able to use HTTP we need some extra tools such as :

- **SSL handshake** : used to authenticate the server and to agree on encryption keys and algorithms.
- **SSL change cipher spec** : selects agreed key and encryption algorithm until further notice.
- **SSL alert** : transfers information about failures.

A TLS packet is obtained through a fragmentation of the PDUs, in turn compressed, with the addition of a keyed MAC, then encrypted and finally the addition of an header that indicates the application protocol in use. A VPN realized with SSL brings to two models :

- **SSL Portal VPN** : we have a gateway that realizes the network, which is accessible from a Web browser.

- **SSL Tunnel VPN** : we access the VPN from the gateway, and then we access from there all network features.

Most SSL VPNs offers one or more core functionalities such as : **Proxying** with which an intermediate device appears as true server to client, **Application Translation** to convert information from one protocol to another, **Network Extension** to provide a partial or complete network access to remote users, typically via Tunnel VPN. The types of services that grants SSL are the following :

- **Authentication** : because we are using certificates.
- **Encryption** : via the user of the SSL/TLS protocol.
- **Access control** : because we can also enforce some types of access regulation when we have the client authentication.
- **Endpoint security controls** : we can also enforce some controls on the endpoints, for example to be sure that it doesn't have an old version of the OS.
- **Intrusion prevention** : it's used to evaluate decrypted data for malicious attacks.

6.1.1 SSL/TLS Heartbeat

Whenever you establish the TLS, there is a channel established. Now, if the two endpoints don't use the channel for an extended time then you have the session. The idea is that we have an extension that allows to keep this session alive, simply using a timer that expires when there is no traffic exchange in the channel. Whenever there is this timer, we exchange a small **Heartbeat packet**, which avoid the re-negotiation of the security parameters for that session. In particular, we introduce two types of packets : **Heartbeat Request** and **Heartbeat Response**. Whenever an endpoint receives an Heartbeat Response simply reset the timer. As a protection against a replay attack, Heartbeat Request packets include a payload that must be returned without change by the receiver in its Heartbeat Response packet. The problem with this mechanism is the following : in the implementation of the answer of the response, happen that the message was placed in memory, it simply read the length of the payload and it get from the memory that amount of bytes from the memory where this packet was stored. If an attacker manages to send an Heartbeat Request with a payload

size much greater than the original one, then he's able to get some bytes from the receiver's memory, that may contains sensible information. This bug is also called **Heartbleed bug**.

6.2 Device placement

Another important point is the place supposed to be where the VPN device is located, since it affects the VPN features such as security, functionality and performance. We can place a VPN device in the following places :

- **Firewall** : whenever the traffic reach the firewall become unencrypted and can reach the internal network or the DMZ. We have no holes in the firewall between external VPN device and internal network, the traffic between device and internal network must go through the firewall and it's very simple to configure. The main disadvantages are the following : limited to VPN functionality offered by firewall vendor, adding new VPN functionality to the firewall can introduce vulnerabilities, external hosts can access the firewall via port 443 (since it must be open on the firewall external interface to allow clients to initiate a connection).
- **Internal network** : the traffic goes in the firewall and reaches the internal VPN device. Also in this case we haven't holes in the firewall between the VPN device and internal network, the VPN traffic is protected from attacks by machines in DMZ since it's encrypted until reaches the VPN device in the internal network, it's very easy to configure. The problems here are that the VPN traffic isn't analyzed by the firewall, the network is compromised if the VPN device is compromised.
- **Single-interface VPN device in DMZ** : it's one of the most common way to realize a VPN. This is useful because we have the possibility to explore the traffic that is outside the VPN using intrusion detection system, but it's a bit more complex to be configured in the firewall, and we have unencrypted traffic in the DMZ.
- **Dual-interface VPN device in DMZ** : in this case the VPN device within the DMZ can be directly connected to the internal network passing through the firewall. Now the unencrypted traffic to internal hosts is protected from other hosts in the DMZ, but it's more complex to be configured in the firewall because we have different source of traffic from the DMZ.

6.3 IPsec

IPsec with respect to SSL VPN is placed on the network layer. It's a collection of protocols that assist in protecting communications over IP networks. IPsec protocols work together in various combinations in order to provide protection for communications. The three main IPsec protocols are :

- **Authentication Header (AH)** : it provides integrity protection for packet headers and data, as well as user authentication, adding its own header. It can optionally provide replay and access protection. AH cannot encrypt any portion of packets. AH can authenticate portions of packets that ESP cannot. AH has two modes :
 - **Transport mode** : AH doesn't create a new IP header. It authenticate IP payload and selected parts of IP header and IPv6 extension headers.
 - **Tunnel mode** : AH create a new IP header for each packet. It authenticate the whole inner IP packet and selected parts of outer IP header and outer IPv6 extension headers.
- **Encapsulated Security Payload (ESP)** : it provides encryption for packet payload data and optionally authentication, adding its own header and trailer. ESP also has two modes :
 - **Transport mode** : ESP uses the original IP header, instead of creating a new one. It encrypt IP payload + any IPv6 extension headers after ESP header.
 - **Tunnel mode** : ESP creates a new IP header for each packet. It encrypt inner IP packet.
- **Internet Key Exchange (IKE)** : its purpose is to negotiate, create and manage **security associations (SAs)**. SA is a generic term for a set of values that define the IPsec features and protections applied to a connection. Each SA is identified by a Security Parameters Index (SPI), which is a 32-bit integer chosen by sender, that enables receiving system to select the required SA, a destination unicast address and a Security Protocol Identifier (AH or ESP).

Often AH and ESP are used together to provide both authentication and encryption. There are several combinations of AH + ESP, for example we apply first AH in transport mode and then ESP in tunnel mode, or both in tunnel mode and so on so forth. TLS is much more flexible respect to IPsec,

because is in the upper levels. TLS also provides application end-to-end security (best for web applications, for example based on HTTPS). IPsec has to run in kernel space and thus is more complex to manage.