PND

Trasciatti

February 8, 2022

0.1 Ethernet and switches

ARP: MAC table for hosts

CAM: MAC table for switches.

0.1.1 MAC addresses

MAC multicast addresses exist, and they're the one that have 1 as least significant bit in the first octet. Broadcast MAC are the one composed by all Fs.

0.2 Ethernet vs IP addressing

ethernet has physical addresses, virtually unchangeable; they aren't a way to tell where the hosts resides. IP has logical addresses that you can change; they are used both to identify and to locate hosts.

0.3 Routers and routing tables

The routing table contains pairs network/next-hop and is used to select the next hop when forwarding an IP packet. It contains the networks directly connected, those advertised with routing protocols and those added statically. **A packet is forwarded to the next hop with the longest prefix matching** when XORing the destination IP with the netmask of each network of the routing table.

0.4 Terminology

0.4.1 General terms

Audit: to monitor against a standard.

0.4.2 MAC in networking security

In cryptography, a MAC (message authentication code) is a short piece of information used for authenticating a message. It's used to confirm that the message came from the stated sender (its authenticity) and has not been changed. The MAC value protects a message's data integrity, as well as its authenticity, by allowing verifiers (who also possess the secret key) to detect any changes to the message content.

0.4.3 Integrity, authentication, etc

Data integrity is used to state the fact data hasn't been modified by someone in the middle of a destination.

1 IP

1.1 IPv4

IPv4 defines IP addresses with 32 bits organized in four octets. A subnet mask defines the boundary between the network and the host portion. Three types of IP addresses:

1. Unicast: refers to a single destination host
2. Broadcast: refers to every host in a network
3. Multicast: refers to a group of addresses in a network

Different classes.

1. Class A: 24 bits for the host (/8 Subnet Mask); from 0.0.0.0 to 127.255.255.255
2. Class B, /16: 128.0.0.0 to 191.255.255.255
3. Class C, /24: 192.0.0.0 to 223.255.255.255
4. Class D, multicast addresses 224.0.0.0 to 239.255.255.255
5. Class E, rerved: 240.0.0.0 to 255.255.255.255

In general, for each network address, the first host address is reserved for the network, and the last is reserved as the broadcast address; e.g. 192.168.8.0/24 is the network address (notice the host portion as 0), and 192.168.8.255/24 is the broadcast address for that network.

The network portion can be also specified with the dotted mask, e.g. for the network 10.10.10.0/26, the notation is 10.10.10.0/255.255.255.192.

RFC 1918 defines the private IPv4 addresses (addresses unique only in the local network); they are the equivalent of IPv6 Unique Local addresses.

RFC 1918 name	IP address range	Number of addresses	Largest CIDR block (subnet mask)	Host ID size	Mask bits	*Classful* c
24-bit block	10.0.0.0 – 10.255.255.255	16 777 216	10.0.0.0/8 (255.0.0.0)	24 bits	8 bits	single class A
20-bit block	172.16.0.0 – 172.31.255.255	1 048 576	172.16.0.0/12 (255.240.0.0)	20 bits	12 bits	16 contiguou
16-bit block	192.168.0.0 – 192.168.255.255	65 536	192.168.0.0/16 (255.255.0.0)	16 bits	16 bits	256 contiguo

Figure 1: IPv4 private addresses

1.1.1 DHCP procedure

1. Host broadcasts DHCP discover
2. DHCP server responds with DHCP offer
3. Host requests IP address with DHCP request
4. DHCP server sens address: DHCP ACK

Note: the address offered is in a variable called yiaddr.

1.2 Ipv6

IPv6 has addresses of 128 bits.

2 TCP/UDP

2.1 TCP

Services that rely on TCP:

1. FTP on port 20, 21
2. SSH on port 22
3. Telnet on port 23
4. SMTP on port 25
5. HTTP on port 80
6. IMAP on port 143
7. SSL on port 334
8. HTTPS on port 443

2.2 UDP

Services that rely on UDP:

1. DNS on port 53
2. DHCP on port 67, 68
3. TFTP on port 69
4. SNMP on port 161
5. RIP on port 520
6. IKEv2 on ports 500, 4500

3 Capture packets

To catch packets that flow in a network we need a traffic dump tool, like dumpcap, wireshark/tshark, and tcpdump. They're all based on the pcap library. All of thm can visualize and save captured data, but only wireshark and tcpdump can also analyze it (decode).

Alternative ways to capture traffic info are: netflow, for statistics and monitoring; and zeek, a framework for traffic inspectiong and monitoring. In these two, the traffic is represented as connections.

To capture packets we will need filters, and there are two types of them: display filters and capture filters. Capture filters limit the amount of network data that is getting saved, and display filters is used to inspect only the packets we want to analyze after they have already been stored.

To actually be able to capture packets, we can use: prmiscuous mode (not very effective on switches), physical taps, port mirroring, ARP cache poison (the use of ARP replies to steal IPs), MAC flooding (fill the CAM tables so the switch act as an hub), DHCP redirection (exhaust the IP addresses of the pool then pretends to be the new DHCP of the network), and redirection and interception with ICMP (ICMP redirect is used).

3.1 Netflow

Is a suite of tools. nfcapd to capture and save netflows; nfdump to analyze netflow files (tcpdump-stye); and nfsen, graphical tool to access captured netflows that uses nfdump as a backend.

3.2 Wireshark

Wireshark processes a packet layer by layer, starting from the bottom of the network stack: it firstly reads the outer frame, extracts the value of the inner protocol and tries to match each field of the inner PDU. The process is repeated for every layer up to the top of the stack.

The promiscuous mode (or monitor mode) is required to capture packets that are not intended to be saw from the host.

Protocols can be detected in two ways: directly, if a frame has a field specifying the protocol; indirectly using tables of protocol/ports combinations and heuristics. This latter doesn't work when protocols use non-standard ports.

3.2.1 Read wireshark hexdump

The hexdump represents each byte (8 bit) with two hexadecimal numbers. So if for example have a 16-bit number as: 00000000'01010100, we can write it as: 00-45.

4 Prevent packet capture

4.1 Dynamic address inspection

It's implemented in switches: DAI validates ARP packets.

4.2 DHCP snooping

It's implemented in switches; it distinguishes between trusted and untrusted ports and uses a database of IP-to-MAC.
Ports that exhibits rogue behaviour can also be automatically placed in a disabled state.

5 IPv6

In general: more addresses, stateless auto-configuration, end to end reachability without NAT and private addresses; better support for mobility, peer to peer networking. Also, services like QoS and VoIP are more robust.

5.1 Addresses

5.1.1 Global unicast addresses

GUA addresses are unique and routable, and their network portion goes from 2000::/3 to 3FFF::/3.

Addresses are in general long 128 bits, and in GUA 48 bits are dedicated to the global routing prefix (network address), 16 are used for the prefix length, and the other 64 are used for the interface ID (like the host portion in IPv4).
We can visualize it as the 3 1 4 rule; 3 groups of 16 bits are the global routing prefix, one group is the subnet ID, and the last 4 are the interface ID; e.g. : (2001:0DB:CAFE):(0001):(0100:208A:0010:0010).

5.1.2 Link local unicast

Link local unicast addresses are unique only on the link; any IPv6 device must have one. They are created either manually or automatically, and usually the first 10 bits are FE80; the interface ID may be random 64 bits, or can be made with the EUI64 format (two parts: OUI and the device identifier are used, and FF-FE bits are plaed between them). E.g : FE99:47FF:FE75:C3E0.
These addresses are used as address before a device gets one dynamically; router's link-local is used as default gateway. Routers also uses them to exchange routing messages, and as the next hops in the routing tables.

5.2 Neighbor discovery

The neighbor discovery protocol encloses diverse functions that may be host-to-router, host-to-host, or redirection (used by routers to inform hosts about a better **first-hop** on the path to the destination). Notice that like ICMP, ND is a messaging protocol. It doesn't implement a single specific function but rather a group of activities that are performed through the exchange of messages.

5.2.1 Host-router functions

These functions are: router discovery (locate the router), prefix discovery, parameter discovery (discover certain parameters of the network such as the MTU), and address autoconfiguration.

Routers send regularly router advertisement messages, to share information about the router itself and the parameter of the network. These messages contains: the suggested TTL to use in the network, the router lifetime (how long use this router as the default one), reachable time (how long, hosts should consider a neighbor to be reachable after they have received reachability confirmation), retransmission time (how long a host must wait before re-transmit a negihbor solicitaion message); lastly, they may contain the following 3 options: source link address (the layer 2 address of the router), MTU, nad prefix information (which prefix/prefixes must be used in the network).

Routers also listen for router solicitation messages, to which they must immediately respond with a router advertisement message. Router solicitaion messages are sent on the multicast IPv6 addresses for all routers, and it include the source MAC of the sender.

5.2.2 Host-host functions

These functions includes: address resolution (determinate the layer 2 address using layer 3), next hop determination, neighbor unreachability detection and duplicate address detection.

The first task that any host must perform when it wants to send a datagram is Next-Hop Determination. This is the process by which a device looks at the destination address in a datagram and decides whether it can be directly delivered (using the address in the message), or must be indirectly delivered (so it will be sent to one of the routers); notice that this is not done for every datagram, instead a destination cache is used.

When a host wants to get the layer two address of a datagram destination, it sends a Neighbor Solicitation ICMPv6 message containing the IP address of the device whose layer two address it wishes to determine. That device responds back with a Neighbor Advertisement message that contains its layer two address. Instead of using a broadcast that would disrupt each device on the local network, the solicitation is sent using a special multicast to the destination device's solicited-node address (The solicited-node multicast address is a special mapping that each device on a multicast-capable network creates from its unicast address). Notice that the target host may be a router.

Devices do not routinely send Neighbor Advertisements the way routers send Router Advertisements. There really isn't any need for this: neighbors don't change much over time, and resolution will occur naturally over time as devices send datagrams to each other. In addition, having advertisements sent regularly by so many devices on a network would be wasteful.

A host may, however, send an unsolicited Neighbor Advertisement under certain conditions where it feels it is necessary to immediately provide updated information to other neighbors on the local network. A good example of this is a hardware failure—in particular, the failure of a network interface card. When the card is replaced, the device's layer two (MAC) address will change. Assuming this can be detected by the device's IP layer, it can send out an unsolicited Neighbor Advertisement message to tell other devices to update their resolution caches with the new MAC address.

As for neighbor unreachability detection, each host (and router) keep a cache with a timer that indicates if a neighbor is reachable. The timer is set (and reset) everytime the host receive a datagram from the corresponding neighbor. A host can also dynamically seek out a neighbor if it needs to know its reachability status. It sends a Neighbor Solicitation to the device and waits for a Neighbor Advertisement in response. It then updates the cache accordingly. The lentgh of the timer depends on the indications of the router, and is communicated with router advertisements.

To avoid the use of duplicate addresses, a host may send a Neighbor Solicitation message to the address he wishes to use. If a Neighbor Advertisement is received in reply then the address is already in use. For this process, the neighbor solicitation is sent to a special multicast address, the solicited node multicast. This address is: multicast, so the host is contacting multiple neighbors, and contains the last 24 bits that the host wants to use. Each host has many solicited node addresses, usually one for every unicast or anycast addresses he has (link-local included), and they are built using the well known prefix ff02::1:ff00:0/104, and appending to it the last 24 bits of the corresponding unicast (or anycast) address. In this way, our host, who wants to know if his address is already in use, may use its last 24-bits to build a multicast solicited node address to send a message to every host that uses that same address; if it receive no response (neighbor advertisement), then no one uses that unicast address.

5.3 Address configuration

There are three main ways an host can obtain a global unicast address (GUA): stateless mode, stateless mode + DHCPv6, or full DHCPv6; a router may obtain its internet-facing-interface ID in these same way, but must use DHCPv6 Prefix Delegation to obtain addresses for its subnetwork(s).

5.3.1 Stateless autoconfiguration (SLAAC)

The SLAAC process to obtain an address work this way: an host send a router solicitation multicast (to all router) message to obtain the information needed to create the address and navigate in internet, so the network prefix, its length, and the DNS address (but of course router advertisement messages contains other useful information, such as the router link-local address). Once the router replies with

the router advertisement, the host will create its address creating an interface ID with a random process or EUI64, and combining it with the network prefix.
The host must also be sure that its address is local in the local network, so he send a neighbor solicitation message with this newly created address; if no one responds, then it means that the address is unique and can be used.

5.3.2 Stateless autoconfiguration + DHCPv6

In this process the host will create its own IPv6 address exactly as in the SLAAC method, but once created, he will use it to obtain information from DHCPv6, such as the DNS name and prefix; this is because these info were not inside the router advertise message, so the host must obtain them in another way.

5.3.3 Stateful DHCPv6

In this method the host will ask for a GUA address directly to a DHCPv6 server; it's similar to the IPv4 version. The host will however send a router solicitation message to obtain the router's link local address to use it as the default gateway. To contact the DHCPv6 server, the host will send a multicast message.

5.3.4 Stateful DHCPv6 Prefix Delegation

DHCPv6 PD is the method used by the router to obtain addresses for its local subnetwork(s). Basically the router may configure its interface ID addresses for upstream routers as it prefers (SLAAC, SLAAC + DHCPv6, or stateful DHCPv6), but must ask a prefix for local subnetwork to a DHCPv6 server.
The messages used for this exchange are the DHCPv6 PD request and DHCPv6 PD reply; once the router has obtain the (sub)network prefix, it can send it via router advertisement.

5.4 Temporary addresses

It's not unusual for hosts to have multiple GUA temporary addresses; there are more than 1 so connections may continue uninterrupted while the address changes. They have a short lifetime and the interface ID is randomly generated.

5.5 Recap on the addresses type

Here's a recap that can help recognize the type of the IPv6 address by looking at its numbers.

GUA addresses: they have the 3-1-4 rule, so the first 3 hextets are for the network ID, 1 hextet is for the subnet ID, and the last 4 for the host; notice that the first network hextet goes from 2000 to 3FFF. This mean these addresses range from 2000::/64 to 3FFF:FFFF:FFFF:(... all Fs)/64.

GUA with EUI64 interface ID: the EUI64 process adds an FF:FE rpughly in the middle of the interface ID portionn of the address, so it's fairly recognizable.

Link-local addresses: the first 10 bits go from FE80 to FEBF (the fixed part is the first 10 bits, with value 1111 1110 10)

Multicast addresses: their first 8 bits are all 1s (so they start with FF), then there are 4 bits that represent a flag, 4 that represent a scope, and the remaining 112 are the group ID. The scope may indicate that is interface-local (1), link-local (2), site-local (5), organization-local (8), or global (E). The flag represent the fact that the address is well known and permanent (0), or temporary assigned (1). For example, FF18:cafe::1 is a temporary organizational multicast address.
Moreover, the group ID portion also indicates the type of the device(s): *pic of table*

Solicited node multicast address: these addresses, used in the neighbor discovery protocol (in particular for the duplicate address detection), are created taking the last 24-bits of a unicast or anycast address and appending them to the following address: ff02::1:ff00:0/104 (basically they are appended

after the rightmost FF).

6 Network security and firewalls

A firewall is a device that beside acting as a router, also contains and implement rules to determine whether packets are allowed to travel from one network to another.
In general, t protect a network, one can: regulate the traffic, protect the traffic with encryption, monitor the traffic and the hosts for suspicious behaviour. The choice of which one apply will depend on the security policy to be fulfilled.

6.1 Host based packet filters

It is the kind of firewall that disciplines the traffic in and out a single host; it specifies the packets that can be received or sent. Some applications: windows firewall, iptables.

6.2 packet filters (in general)

Stateless packets filters make some assumptions: there's a policy that states what is allowed and what is not; bad and good traffic is identifiable by ip-address, port number, etc.; and the firewall itself is immune to penetration.
They operate on datalink, network and transport layer (not fully since they have no context).
In general, rules are checked from top to bottom, the first matching rule is applied, and one implicit rule is applied if no one applied (usually drop).
How to know who started communication? In some easy case we can check the flag. But the payload of the transport layer cannot be inspected, so in most cases is impossible to know.
Another big limitation is that there are no authentication facilities.

6.3 Stateful packet filters

Stateful inspection firewalls can keep track of established connections; they can drop packets based on their source or destination IP address, port numbers and TCP flags (remember that stateless filters know these flags too).

6.4 Network Access Control List

Screening routers based on ACL list the rights for accessing and using the network. They distinguish between in/out traffic, per interface/port. It is **stateless**, meaning that each packet is treated independently, without any knowledge of what has come before.

6.5 Bastion host

A bastion host is a hardened computer that deals with outside traffic to protect a network. Hardening is the task that removes vulnerabilities in a computer system, such as shutting down unused/dangerous services, strengthening access control list on vital files, remove unnecessary accounts and permissions, and use a stricter configuration for vulnerable components such as DNS, FTP, Apache, Tomcat ecc..
They are usually used as application proxy gateways.

6.6 DMZ

Is a computer or a small network inserted as neutral zone between a company's private network and the outside public traffic network. It provides secure segregation of network that host services for users, visitors o partners.
It reduces and regulates the access to internal components of an IT system, and provides a defense-in-depth approach to security.

6.6.1 defense-in-depth

It's a security approach in which IT systems are protected using multiple overlapping systems; it adds redundancy to defensive measures, aim to remove single point of failures, and finds the right balance between complexity and multiplicity of defense mechanisms.

6.7 Other types of firewalls

Proxies are discussed in depth in their own section; they are just mentioned here.

6.7.1 Application-level filtering (proxy)

Proxies have a separate special purpose mechanism for each application (mail, HTTP, FTP, etc.); they support user-to-gateway authentication: you can log into the proxy server with username and password.
Proxies may introduce lag, and are not always transparent.

6.7.2 Circuit-level gateways (generic proxy)

Also known as TCP relays, they relay the connections of connected hosts in a protocol independant manner; they also provide user-authentication, and usually don't do content filtering. SOCKS is the sandard; it also works with UDP, and operate in the session layer (of the OSI, in the TCP/IP stack is simply an application). Another example is TOR.
These proxies splice and relay TCP connections without examining the contents; they can use an ACL (do packet filtering), and have less control than application-level gateways.

6.7.3 Host based firewalls

A personal firewall that protect is own host. It enable specific services and ports that will be used to send and receive traffic; it also protect the entire system if a process is compromised.

7 Iptables

It's the implementation of packet filtering firewall for linux that runs in kernel space; t inserts and delete rules from the kernel's packet filtering table. It can also operate at the transport layer, seeing TCP/UDP flags and sequence number. It can keep track of streams, and is able to implement SNAT and DNAT. Basics of iptables:

1. Rules are grouped in tables
2. Each table has different CHAINS of rules
3. each packet is subject to each rule of a table
4. packet fates depend on the first matching rule

There are three principal built-in rule chains: input, output and forward. Input chain rules are for packets that are meant to be received by the host; output chain are for packets that the host is sending, and the forward chain is for packet that are only forwarded by the host and not meant to be received. Two more chains are PREROUTING and POSTROUTING. The first one is to apply to packets before the routing decision has been made, and the second is applied after the routing decision is made.

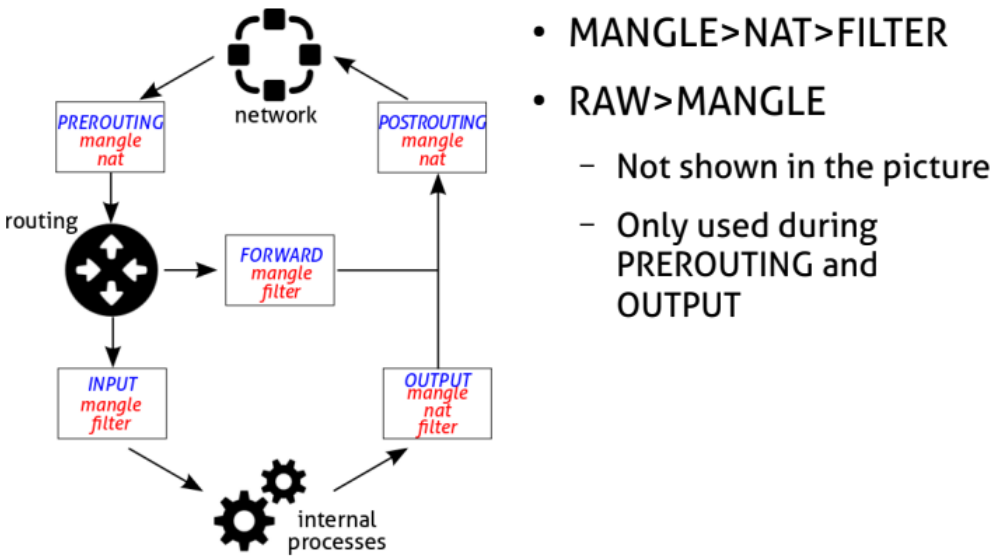


Figure 2: Iptables chains

If a packet reaches the end of a chain, then is the chain policy to determine the fate of the packet (drop/accept).
There are 4 targets: accept, drop, reject (drop but send an error message to the source host of the blocked packet), and log. Log send the packet to the sylog daemon for logging; one can't drop and log at the same time, once the log is done, then iptables continues processing the packet with the next rule in the table.

7.1 Tables

There are 4 built-in tables: MANGLE: manipulates bits in TCP header; FILTER: acket filtering; NAT, and RAW: excetions to connection tracking. When present, RAW table has the highest priority, and is not loaded by default.

7.1.1 MANGLE

It's used for IP header manipulation; it should not be used for NAT and for filtering. Five chains: PREROUTING, INPUT, OUTPUT, FORWARD, POSTROUTING.

7.1.2 FILTER

It's used to filter out packets; three chains: INPUT, OUTPUT, FORWARD **(but only if the host is configured as a router)**.

7.1.3 NAT

Used to translate the packet's source filed or destination. Only the first packet in a stream will hit this table, the others will automatically have the same action.
This table has special targets: DNAT, SNAT, MASQUERADE (dynamic nat, when fw interface address is dynamically assigned), REDIRECT (redirect the packet to the machine itself).
It has the following chains: PREROUTING, POSTROUTUTING, OUTPUT.
In particular, the DNAT target is used in the PREROUTING chain, the SNAT is used in the POSTROUTING, and smilar the MASQUERADE is used in the POSTROUTING.

7.2 More on chains

It's possible to specify a jump rule to a different chain within the same table; if the end of this new chain is reached,then the packet is sent back to the invoking chain.

7.3 iptables logging

LOG is a possible target, but a non terminating one (the packet will continue to flow to the next rule); to log dropped targets, we use the same rule we use to drop the packet, but with the LOG target (and subsequently DROP with the same rule). The log is printed via kernel log, and it can beread with dmesg or syslogd.
It's possible to specify the pe of log (emerg, alert, crit, notice, info, debug, err) with –log-level 'level'; and add further information in front of all messages produced by the logging action with –log-prefix 'prefix' (e.g. 'invalid packet').

8 NAT

8.1 Routed vs Transparent firewall modes

In routed mode, a firewall is a hop in the routing process; it IS a router responsible of its own internal networks). In transpant mode, a firewall works with data at layer 2 and is not seen as a router hop to connected devices; it can be implemented using bridged NICs.

8.2 Routable network addresses

Routable addresses need to be unique on the internet to be publicly reachable (for this reason they are called public address). Non routable addresses are special purpose addresses such as: private addresses (prefix 10/8,, 172.16/12, 192.168/16), loopback addresses (127/8), shared addresse space (100.64/10)

8.3 Features of NAT

NAt translates addresses to allow a LAN with private addresses to connect outside. In a routed firewall, it can filter requests from hosts from WAN directed to LAN; it allows hosts requests from the LAN side to reach the WAN side, and does not expose LAN hosts to external port scans.

8.4 Goals of NAT

With NAT, a private network can use just one IP address provided by the ISP to connect to the internet, while the hosts use private addresses. It makes possible to change the addresses of the devices inside a private network without notifying the outside world; it also makes possible to change the ISP without changing internal IPs. As a security plus, devices in a private network are not explicitly addressable by the external network, nor visible from the outside.

8.5 Source NAT (SNAT)

SNAT 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. It can also change source TCP/UDP ports in the header.
The firewall/roouter forwards replies from the external server to the client; it enables the client-server session or connection to continue on another port as requested by the external server, forwarding any responses by the server to the client.

8.6 Basic NAT and NAPT

Basic NAT: a bloc of external/public IP addresses are set aside for translating the addresses of hosts within a private domain as they originate sessions to the external domain. The NAPT also translates transport identifiers, like TCP and UDP port numbers, as well as ICMP query identifiers. Basically it multiplexes a number of private hosts into a single public IP mapping hosts with port numbers. Private IP/port number and public IP/port number pairs are stored in the NAPT table. By default, NAPT routers block all incoming ports.

8.7 Destination NAT (DNAT)

Enables servers located inside the firewall/router LAN to be accessed by clients located outside; the service appears to be hosted by the firewall/router. It uses the NAT table to translate incoming packets from the firewall WAN IP address to the internal address of the server. It also forwards the replies to the client requests from the server. It permits to continue the server-client connection on another port as requested by the server.
DNAT is also called port forwarding because according to the port accessed from the external interface, the packets can be forwarded to different internal hosts.

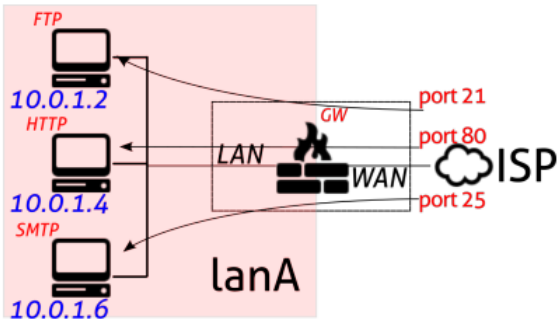


Figure 3: DNAT port forwarding

Change port numbers on DNAT sessions, to enable an internal server to provide a particular service. An example: a host might be configured to provide outgoing SMTP service for the LAN on port 25 and incoming SMTP service on port 2525; then the firewall will translate the port numbering for DNATted incoming SMTP requests from 25 to 2525 and will also translate outgoing responses on this port intelligently.

8.8 NAT pros

1. Simple security due to stateful filter implementation
2. User/application tracking
3. privacy and topology hiding
4. independent control of addressing in a private network
5. global address pool conservation

8.9 NAT unsupported application

1. Applications that have a realm-specific IP address information in payload (e.g. FTP)
2. bundled session applications
3. peer-to-peer applications (how do one peer obtain the private endpoint of the other peer? you can't contact a user if he didn't contact you before, creating a deadlock; but only if both of them are behind the NAT)
4. IP fragmentation with NAPT enroute
5. applications requiring retention of address mapping
6. Applications requiring more public addresses than available
7. Encrypted protocols like IPsec, IKE, Kerberos (because they require a single, unambiguous IP address to identify a single host; also, in transport mode IPSec, the AH and ESP use integrity check that is based on the value of the entire payload; when NAT updates TCP/UDP checksum, the integrity check will fail)

Meanwhile a solution for encrypted application traffic is difficult to elaborate, there's a method to help two peer (both in private networks and using NATs) to communicate: the hole punching. The main idea is to find the public IP address of the other peer and initiante a connection to create a NAT state, so that replies can be corretly passed; Interactive Connectivity Establishment (ICE) is used for this purpose, discovery and help two peers to communicate using STUN and TURN servers. Both of these servers help peers to communicate, with the difference that STUN is needed only to begin the conversation, meanwhile the TURN will be used for the entire length of the conversation (for this reason STUN servers are preferred).

Example of third party for NAT traversal

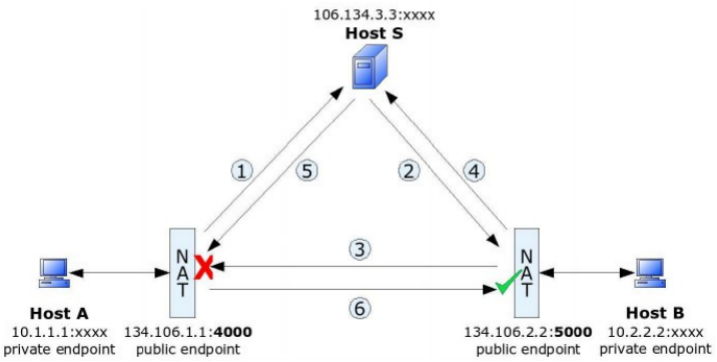


Figure 4: Hole punching with STUN server

Example: UDP hole punching

1)Host A submits the request for communication with Host B to Host S

2)Host S submits the public endpoint (port 4000) of Host A to Host B

3)Host B sends a packet via it's public-private endpoint (port 5000) to the public endpoint of Host A (4000).

1) The NAT system of Host A rejects the packet, but a NAT session on B's side is now established ("a hole is punched"), ready to translate any packets from A's public endpoint (port 4000) to B's public endpoint (port 5000) into B's corresponding private endpoint

4)Host B notifies Host S and awaits an incoming connection from Host A

5)Host S submits the public endpoint of Host B (port 5000) to Host A

6)Host A uses its public-private endpoint (port 4000) to establish a connection with Host B's public endpoint (port 5000)

This technique appears to work with 82 percent UDP and 64 percent TCP NAT systems

B. Ford. *Peer-to-peer communication across network address translators,2005. USENIX Annual Technical Conference*

Figure 5: Details of hole punching

9 Link local attacks

9.1 Network sniffing (or eavesdropping)

It's the practice that captures packets from the network transmitted by other nodes and reading the data content in search of sensitive information. It's done using tools called network sniffers (or protocol analyzers), like etetrcap, bettercap, networkminer, driftnet, dsniff, macof, etc.. It requires to be along the path or in a broadcast domain, and it works passively. The collected data is then analyzed with protocol decoders or stream reassemblers.
To realize network sniffing, the network interface must be in promiscuous mode (stores data even when not meant for them), and the sniffer must be in the same network/on the same path: ideally, in a non-switched LAN (lan with HUBs), because hubs duplicate frames to all ports; in a switched LAN, it's possible to break switch segmentation by flooding the switch with a large amount of frames (MAC flooding), or perform an ARP spoof attack to redirect the traffic from one port to another; in a wireless lAN it's easy if there's no encryption/weak encryption, because the scenario is very similar to the LAN with hubs.

9.1.1 Switch segmentation

Switch are multiport bridges that regenerate a frame only in the segment of destination; it learns the host in each network segment in real time. If a MAC is unknown, then the frame is replicated on all the ports, causing flooding. So it's possible to fill the CAM table of switches, meaning that the switch will start flooding and every frame will be replicated in each port (or the switch can freeze/crash). Today this attack is not really effective since a switch can learn multiple MAC for each port, and upon detecting an invalid MAC it can be rejected or directly block that port.

9.1.2 ARP spoofing

ARP requests work like this: they are broadcasted from an host (who want to know the MAC of a certain IP address) to the entire network; then each computer receives the request and examines the IP address; the computer whose IP corresponds replies to the original host (his MAC was in the ARP request), meanwhile the others simply discard the ARP request message. ARP tables store pairs of IP-MAC; once stored, pairs expire after a timeout in the order of minutes.
Gratuitous ARP responses are used by hosts to announce their IP address to the local network, and to avoid duplicate IP addresses in the network. It's a broadcast message, and routers may use cache information gained from gratuitous ARP responses. There's no guarantee that the ownership of the communicated IP/MAC pair is valid. For example, if a host as an entry for an IP, say 1.2.3.4, and receive a malicious ARP gratuitous response, the already stored MAC will be overwritten. With this technique, it's possible to perform DOS and MITM attacks.

10 VPN

10.1 Principles

A VPN is a virtual network built on top of an existing network infrastructure, which can provide a secure communication mechanism for data and other information transferred between two endpoints. It is typically based on the use of encryption, and there are several possibilities as how and where to perform the encryption, and which part of the conversation must be encrypted.

The security goals for VPNs are: confidentiality of data, integrity of data, peer authentication, replay protection, access control and traffic analysis protection.
The usability goals are: transparency, flexibility and simplicity.

10.2 Types of VPN by layer

10.2.1 Physical layer

Literally new cables.

1. Confidentiality: on cable
2. Integrity: on cable
3. authentication: none
4. Replay protection: none
5. Traffic analysis protection: on cable
6. Access control: physical access
7. Transparency: full transparency
8. Flexibility: can be hard to add to new site
9. Simplicity: excellent

10.2.2 Datalink

This type protects a single link. Must be asked to the iSp (?9

1. Confidentiality: on link
2. Integrity: on link
3. authentication: none
4. Replay protection: none
5. Traffic analysis protection: on link
6. Access control: physical access
7. Transparency: full transparency
8. Flexibility: can be hard too add to new sites
9. Simplicity: excellent

10.2.3 Network

1. Confidentiality: between hosts/sites
2. Integrity : between hosts/sites
3. authentication: for host or siet
4. Replay protection: between hosts/sites
5. Traffic analysis protection: host/site infformation exposed
6. Access control: to host/site
7. Transparency: possible
8. Flexibility: may need HW or SW modifiations
9. Simplicity: good for site to site, not good for host to site

10.2.4 Transport layer

1. Confidentiality: between apps/hosts/sites
2. Integrity: between apps/hosts/sites
3. authentication: between apps7hosts/sites
4. Replay protection: between aps/hosts/sites
5. Traffic analysis protection: protocol/host/site info, exposed
6. Access control: user/host/site
7. Transparency: user and SW tranparency possible
8. Flexibility: HW or SW modifications
9. Simplicity: good for site to site, not good for host to site

10.2.5 Application

1. Confidentiality: between users/apps
2. Integrity: between users/apps
3. authentication: user
4. Replay protection: between apps
5. Traffic analysis protection: all but data exposed
6. Access control: only data access secured
7. Transparency: only user transparency
8. Flexibility: SW modifications
9. Simplicity: depends on application

10.3 Tunneling

The tunneling is the operation of a network connection on top of another network connection; it allows hosts or sites to communicate through another network that they don't want to use directly.

Site-to-site tunneling enables a PDU to be transported from one site to another without its contents being processed by hosts on the route. The idea is to encapsulate the PDU in another PDU sent out on the network that connect the two sites; encapsulation takes place in the edge router on the source site, and decapsulation takes place in the edge router of the destination site. Notice that the IP address in the outer PDU is the IP address of the destination edge router.

A simple tunnel may be realized using tun (encapsulates IP layer) and tap (encapsulates ethernet layer).

10.4 Tunneling for VPNs

Tunneling provides a basic method to implement a VPN; There are several possibilities: where to implement it (router, host, application), and on which layer (network or transport).

10.4.1 Secure socket layer

SSL 3.0 has become the standard for transport layer security; it's placed on top of TCP (so it work with every TCP protocol), and can be implemented over routers (or proxies) to provide a site-to-site tunnel. HTTP on top of TLS (HTTPS) works with symmetric encryption.

The SSL architecture adds an extra layer between transport and application layers, and add extra elements on the application layer. In the extra layer resides the record protocol, who offers basic encryption and integrity services to applications. The extra elements on the application layer are: the handshake, used to authenticate server and to agree on encryption keys and algorithms; the change cipherspec, who selects agreed keys and algorithm until further notice; and lastly the alert, who transfers information about failures.

10.4.2 SSL/TLS record protocol

The protocol applies the following steps:

1. Fragmentation of application data into blocks of $<= 2^{14}$ bytes
2. (optional) lossless compression
3. Addition of a keyed MAC, using a shared secret MAC key
4. Encryption using a shared secret encryption key
5. Addition of header to indicate the application protocol in use

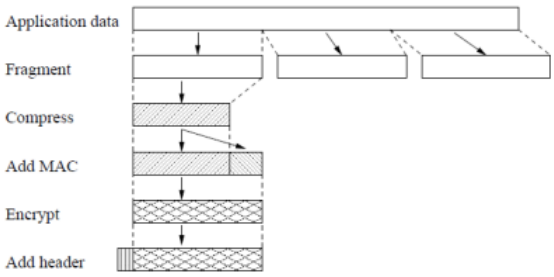


Figure 6: SSL/TLS steps

10.4.3 SSL/TLS handshake protocol

It's a 4-phase client/server (client is just the name of the initiator) protocol to establish parameters of the secure connection.

1. Hello phase: 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 phase: Server executes selected key exchange protocol (if needed). Server sends authentication info. (e.g. X.509 cert.) to Client.
3. Client authentication and key exchange phase: Client executes selected key exchange protocol (mandatory). Client sends authentication info. to Server (optional).
4. Finish phase: Shared secret key is derived from pre-secrets exchange in 2, 3. Change Cipher Spec. protocol is activated; summaries of progress of Handshake Protocol are exchanged (cipher spec and finish) and checked by both parties

10.4.4 SSL/TLS security capabilities

It's expressed by a descriptive string, specifying the version of SSL/TLS, key exchange algorithm, grade of encryption, encryption algorithm, mode of block encryption (if adopted), and cryptographic checksum algorithm. An example: TLS_RSA_WITH_AES_128_CBC_SHA; this means: (latest version of) TLS, RSA (key exchange), AES_128 (128 bit AES encryption), CBC (cipher block chaining), SHA (use HMAC-SHA digest).

Possible ways to agree on secret key exchange are:

1. RSA
2. DHE RSA: ephemeral diffie-hellman with RSA signatures
3. DHE DSS: ephemeral diffie-hellman with DSS signatures
4. DH RSA: diffie-hellman with RSA certificates
5. DH DSS: diffie-hellman with DSS certificates
6. DH anon: anonymous diffie-hellman (no authentication)
7. NULL: no key exchange

Notice that "key exchange" only establishes a pre-secret! From this, a master secret is derived by a pseudo-random function (PRF). Shared secret encryption key is derived by expansion of master secret with another PRF. (In TLS several keys are derived for different purposes).

10.4.5 SSL/TLS Heartbeat

It's a process that allows to keep an established session alive, even after the data exchange between the two endopoints terminates; this is to avoid the re-negotiation of security parameters. It works like this: and endpoint sends a HeartbeatRequest message to the other endpoint, and both start a timer called the retransmit timer. During this time, the sender endopoint cannot send another HeartbeatRequest. An SSL/TLS session is considered to have terminated in the absence of a HeartbeatResponse packet within the time interval. To avoid replay attacks, HeartbeatRequest packets include a payload that must be returned without change by the receiver in the HeartbeatResponse packet. Since the size of the payload to be sent back is specified but not checked, it's possible to set it to the maximum without actually having a payload that long; in this way the receiver will send those max number of byte, starting from where the payload is memorized, possibly sending sensitive information within the paylaod.

10.5 SSL VPN architecture

There are two primary models: the SSL portal VPN, who allows remote users to connect to VPN gateway from a web browser, and to access services from web site provided on gateway, and the SSL tunnel VPN, who has more capabilities than portal VPN and allows remote users to access the network protected by the VPN gateway.

Most SSL VPNs offer one or more of these core functionalities: proxying: intermediate device appears as true server to client (e.g. web proxy); application translation: conversion of information from one protocol to another (e.g. portal VPNs offers translation for application which are not web-enabled, so users can use the web browser to access applications with no web interface); network extension: providing of partial or complete network access to remote users, typically via tunnel VPN. This latest has two variants: full tunneling, where all traffic goes through tunnel, and split tunneling, where only some traffic goes through tunnel (e.g. organization traffic), meanwhile other uses the default gateway.

10.6 SSL VPN device placement

The main options are:

1. VPN functionality in firewall
2. VPN device in internal network
3. Single-interface VPN device in DMZ
4. Dual-interface VPN device in DMZ

10.6.1 VPN-enabled firewall

The VPN device communicates directly with internal hosts. The advantages are: no holes in FW between external VPN device and internal network; the traffic between device and internal network must go through the firewall; simpler network administration since there's only one box to administer.
The disadvantages are: limited VPN functionalities depending on the FW vendor; the FW is directly accessible to external users via port 443 (https port, and of course it must be opened so external devices can start TCP communications).

10.6.2 VPN device inside a DMZ

The advantages are: the internal network is protected against compromised VPN device; traffic between device and the internal network must go through the firewall; the IDS (intrusion detection system) in DMZ can analyze traffic destined for the internal network.
The disadvantages are: numrous ports open in the firewall between the device and internal hosts; decrypted traffic from device to the internal network must be sent through the DMZ; firewall is bypassed when the user traffic is destined for hosts in DMZ. Notic ethat the TCP port 443 is opened on the firewall for the address of the VPN device.

10.6.3 VPN device with two interfaces in DMZ

The VPN device is connected with an interface on the DMZ network, and with the other to the firewall; in this way the device can communicate with internal hosts with the firewall-connected interface, meanwhile external clients use the DMZ interface.
The advantages are: the same that we have when the VPN device is placed into a DMZ; the unencrypted traffic to internal hosts is protected from other hosts in the DMZ; and only the firewall interface connected to the device internal interface need to permit traffic from the VPN device.
The disadvantages are: numerous ports are opened in the firewall between the device and the internal hosts; it also may add introduce additional routing complexity; and the firewall gets bypassed if split tunneling is not used and the user traffic is destined for hosts in the DMZ.

10.7 SSL security

Server may know nothing about clients, unless client-side certificates are used; usually there's no authentication in SSL, only in the application layer. The client knows the server certificate, but it's a mere binding between a name and a key. Can we trust every certificate provider authorities? So the cryptography part of the protocol is fine, but certificates are not exactly trustworthy.

10.8 IPSec

It's a network layer protocol for providing security over IP; it's part of IPv6 and an add-on for IPv4. It can handle all the possible security architectures: host-to-host, host-to-gateway, gateway-to-gateway.

Feature	Gateway-to-Gateway	Host-to-Gateway	Host-to-Host
Protection between client and local gateway	No	N/A (client is VPN endpoint)	N/A (client is VPN endpoint)
Protection between VPN endpoints	Yes	Yes	Yes
Protection between remote gateway and remote server (behind gateway)	No	No	N/A (client is VPN endpoint)
Transparency to users	Yes	No	No
Transparency to users' systems	Yes	No	No
Transparency to servers	Yes	Yes	No

Figure 7: IPSec characteristics over different architectures

10.8.1 IPsec Fundamentals

IPsec grants the following:

1. Data origin authentication: it's impossible to spoof the source/destination address without the receiver noticing; it's not possible to replay a recorded IP packet without the receiver noticing
2. Connectionless data integrity: the receiver is able to detect any modification of IP datagrams in transit
3. Confidentiality: it's not possible to eavesdrop the content of IP datagrams, and also limited traffic flow confidentiality is granted
4. 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

10.8.2 IPsec Architecture

1. Concepts: Security Association (SA), Security Association Database (SAD), Security Policy (SP), Security Policy Database (SPD)
2. Fundamentals Protocols: Authentication Header (AH), Encapsulating Security Payload (ESP)
3. Protocol Modes: transport mode, tunnel mode
4. Key Management Protocols (ISAKMP, IKE, IKEv2)

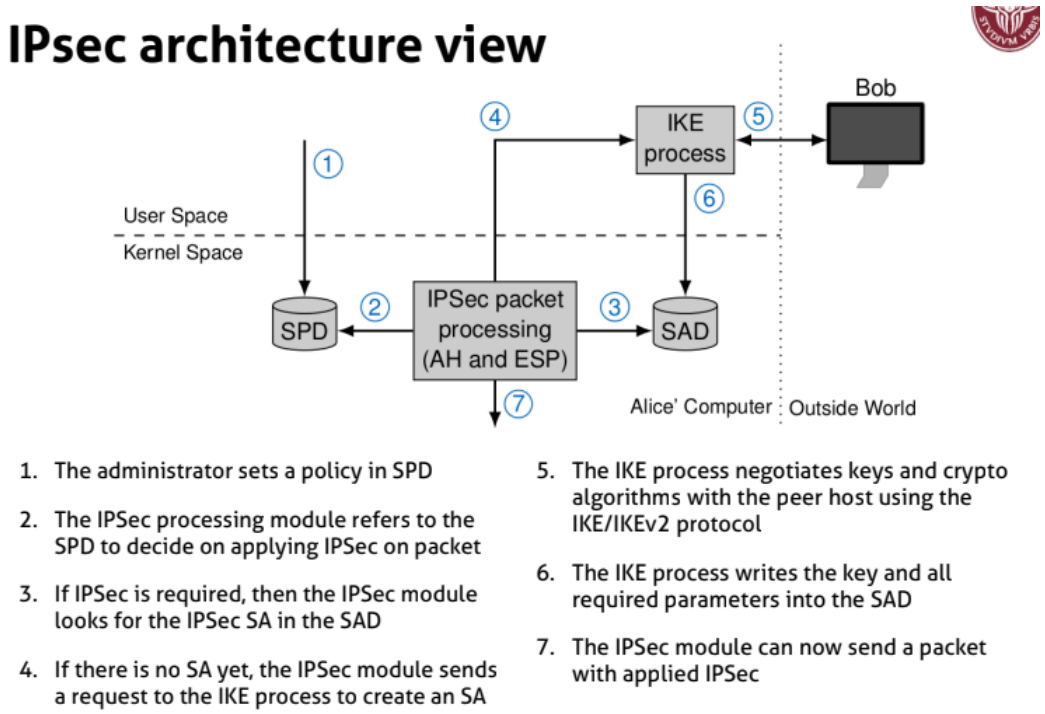


Figure 8: Architecture of IPsec

10.8.3 IPsec security policies

A security Policy (SP) specifies which and how security services should be provided to IP packets; it can be divided in: selectors, who identify specific IP flows; required security attributes for each flow (security protocol like AH or ESP, protocol mode transport or tunnel, and other parameters such as policy lifetime, port number, etc.), and actions: discard, secure or bypass.

Security policies are stored in the security policy database (SPD).

The possible actions of a policy are:

1. Discard: reject to receive / send the packet
2. Bypass(none): do not handle with IPsec
3. Secure(ipsec): handle with IPsec

How to process the packet is in the form: protocol/mode/src-dst/level, where the protocol specifies AH, ESP or ipcomp (for payload compression), mode is either tunnel or transport, src-dst are the end-points of the tunnel (if needed), and the level might be default, use, require or unique; this specifies the level of the SA when a keying daemon is used.

Some example of security policies:

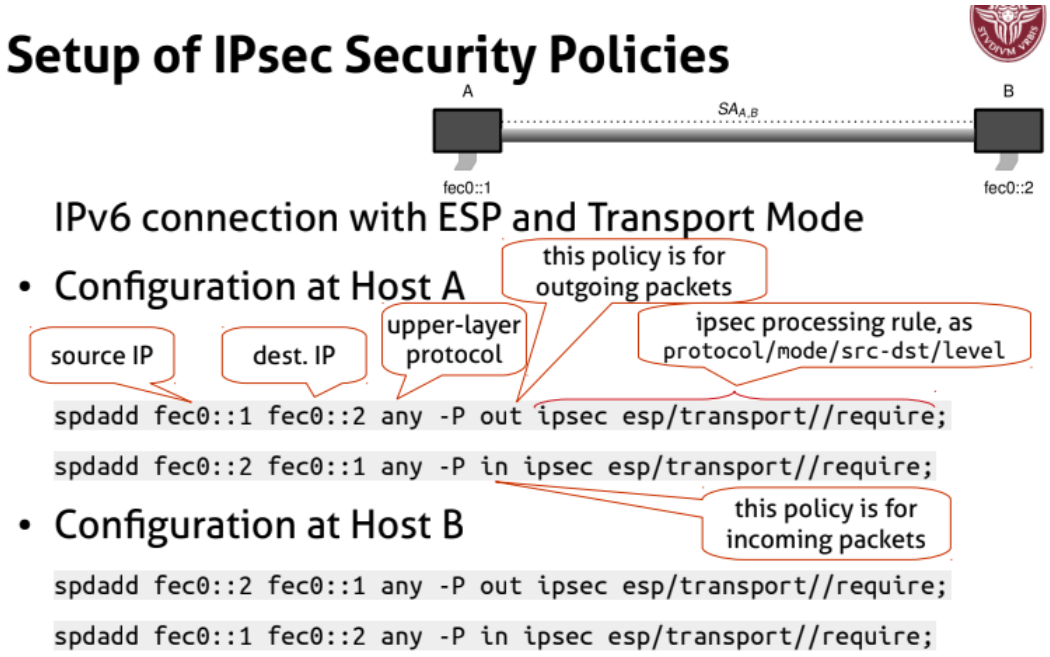


Figure 9: Manually added security policies

Yet another

- Example
  ESP Tunnel for VPN
- Configuration at Gateway A

```
spdadd 10.0.1.0/24 10.0.2.0/24 any -P out ipsec
       esp/tunnel/172.16.0.1-172.16.0.2/require;

spdadd 10.0.2.0/24 10.0.1.0/24 any -P in ipsec
       esp/tunnel/172.16.0.2-172.16.0.1/require;
```

- Configuration at Gateway B:

```
spdadd 10.0.2.0/24 10.0.1.0/24 any -P out ipsec
       esp/tunnel/172.16.0.2-172.16.0.1/require;

spdadd 10.0.1.0/24 10.0.2.0/24 any -P in ipsec
       esp/tunnel/172.16.0.1-172.16.0.2/require;
```

Figure 10: Manually added security policies for two gateways

10.8.4 Security Associations (SA)

A security association is a simplex channel that describes how packets need to be processed. It defines the employed encryption / authentication algorithms and keys; each SA is associated with either AH or ESP, but not both.

Bidirectional communication requires two security associations; they can be setup as: host-to-host, host-to-gateway, gateway-to-gateway.

Security associations are stored in the Security Association Database (SAD), and each one is identified by the SPI, a 32-bit integer chosen by the sender that enables the receiving system to select the required SA .

Some example of Security associations:

Manually setting up an AH SA

```
# basic syntax: add src dst proto spi -A authalgo key;

add fec0::1 fec0::2 ah 700 -A hmac-md5
           0xbf9a081e7ebdd4fa824c822ed94f5226;

add fec0::2 fec0::1 ah 800 -A hmac-md5
           0xbf9a081e7ebdd4fa824c822ed94f5226;
```

- Manually setting up an ESP SA:

```
# basic syntax:  add src dst proto spi -E encalgo key;

add fec0::1 fec0::2 esp 701 -E 3des-cbc
           0xdafb418410b2ca6a2ba144561fab354640080e5b7a;

add fec0::2 fec0::1 esp 801 -E 3des-cbc
           0xdafb418410b2ca6a2ba144561fab354640080e5b7a;
```

Figure 11: Manually added SAs

Notice that destination addresses may be only unicast.

10.8.5 Security Association Database (SAD)

An entry (a SA) is uniquely identified by a Security Parameter Index (SPI). The SPI value is used to map the traffic to the appropriate SA for inbound traffic, and is looked up for outbound SAs.

A SA entry in the SAD includes: the Security Parameter Index (SPI), the IP source/destination address, a security protocol identifier (AH or ESP), the current sequence number counter (for replay protection); protocol algorithms, modes, IVs and keys for authentication and encryption; security association lifetime, IPsec protocol mode (tunnel/transport), and additional information.

10.8.6 Processing of IPsec outgoing traffic

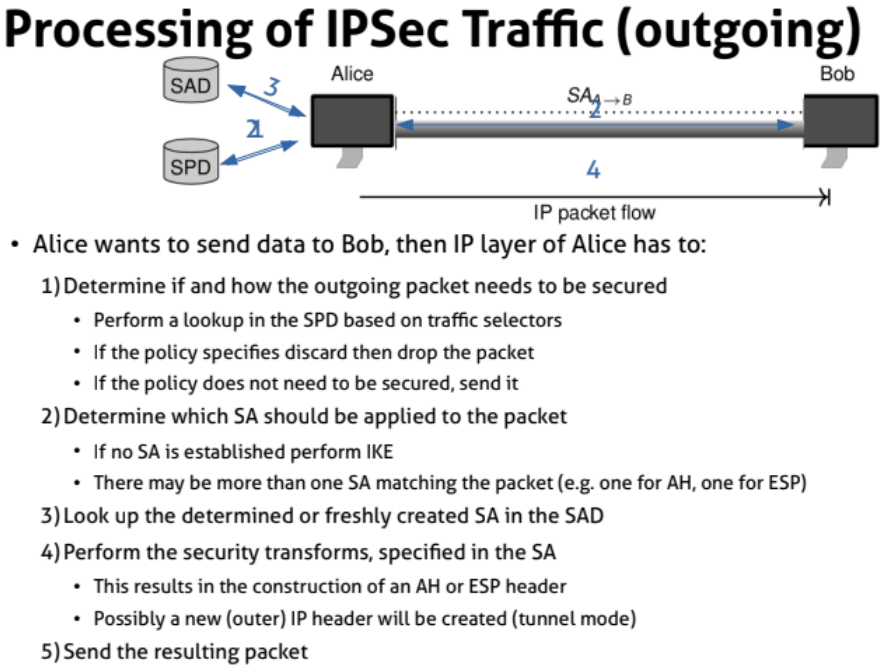
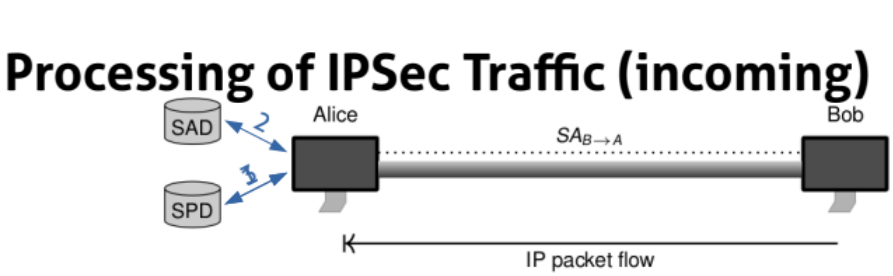


Figure 12: How IPsec works for outgoing traffic

10.8.7 Processing of IPsec incoming traffic



- Alice receives data from Bob, then the IP layer of Alice has to:

1) If packet contains an IPSec header
   - Perform a lookup in the SPD, if Alice is supposed to process the packet
   - Retrieve the respective policy
2) If Alice is supposed to process the packet
   - Extract the SPI from the IPSec header, look up the SA in the SAD and perform the appropriate processing
   - If there's no SA referenced by the SPI ⇒ Drop the packet
3) Determine if and how the packet should have been protected
   - Perform a lookup in the SPD, evaluating the inner IP header in case of tunneled packets
   - If the respective policy specifies discard ⇒Drop the packet
   - If the protection of the packet did not match the policy ⇒Drop the packet
4) Deliver to the appropriate protocol entity (e.g. network / transport layer)

Figure 13: How IPsec works for incoming traffic

10.8.8 IPsec services

Services are provided by the following separate sub protocols: authentication header (AH), who gives support for data integrity and authentication of IP packets, encapsulated security payload (ESP), who gives suport for encryption and (optionally) authentication, and internet key exchange (IKE), who gives support for key management.

Service	AH	ESP (encrypt only)	ESP(encrypt+authent.)
Access Control	+	+	+
Connectionless integrity	+		+
Protection between VPN endpoints	+		+
Data origin authentication	+		+
Reject replayed packets		+	+
Payload confidentiality		+	+
Metadata confidentiality		partial	partial
Traffic flow confidentiality		(*)	(*)

Figure 14: Services of IPsec and what they provide

10.8.9 IPsec modes

It can work in transport mode and in tunnel mode. In transport mode, it provides protection for a transport layer packet embedded as payload in an IP packet. In tunnel mode, it provides protection for an IP packet embedded as payload in another IP packet.

	Transport Mode SA	Tunnel Mode SA
AH	Authenticate IP payload and selected parts of IP header and IPv6 extension headers.	Authenticate entire inner IP packet and selected parts of outer IP header and outer IPv6 extension headers.
ESP	Encrypt IP payload + any IPv6 extension headers after ESP header.	Encrypt inner IP packet.
ESP + authent.	Encrypt IP payload + any IPv6 extension headers after ESP header. Authenticate IP payload.	Encrypt and authenticate inner IP packet.

Figure 15: Services of IPsec in different mode

10.8.10 Authentication Header (AH)

Both ESP and AH can be applied at the same time with different ordering, and if ESP is applied first, then AH is the outer header; with this setup ESP is also protected by AH, but then two SAs (one for each of AH/ESP) are needed for each direction.

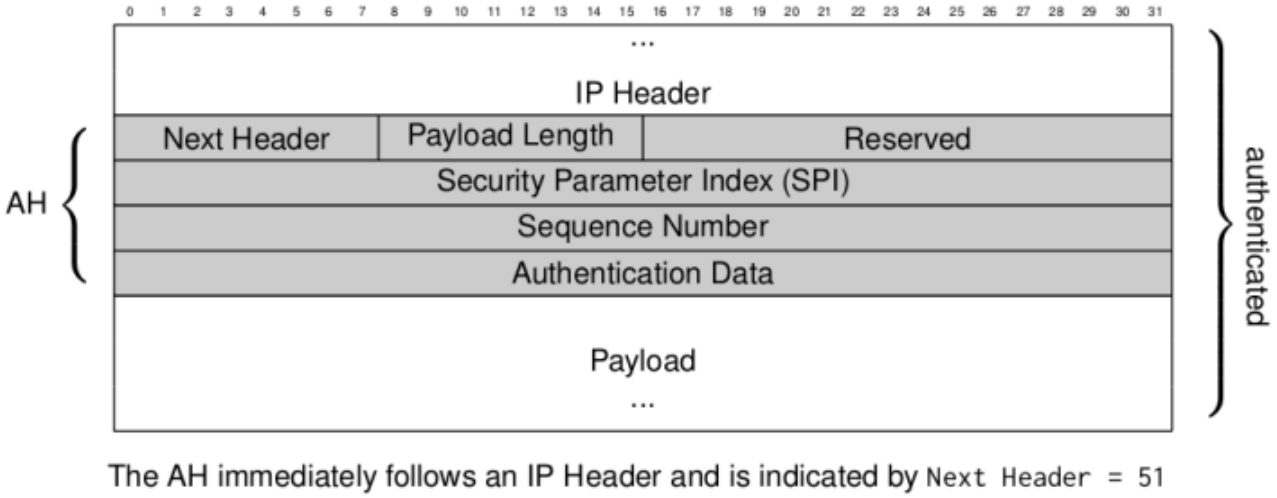


Figure 16: The authentication header

Although the AH protects the outer IP header, some of its fields must not be protected, as they are subject to change during transit (this applies to mutable IPv4 options and IPv6 extensions); such fields are assumed to be zero for MAC computation. They are (for IPv4): the type of serve, flags, fragment offset, time to live, and header checksum.

10.8.11 Encapsulating Security Payload (ESP)

Both ESP and AH can be applied at the same time with different ordering, and if ESP is applied first, then AH is the outer header; with this setup ESP is also protected by AH, but then two SAs (one for each of AH/ESP) are needed for each direction.

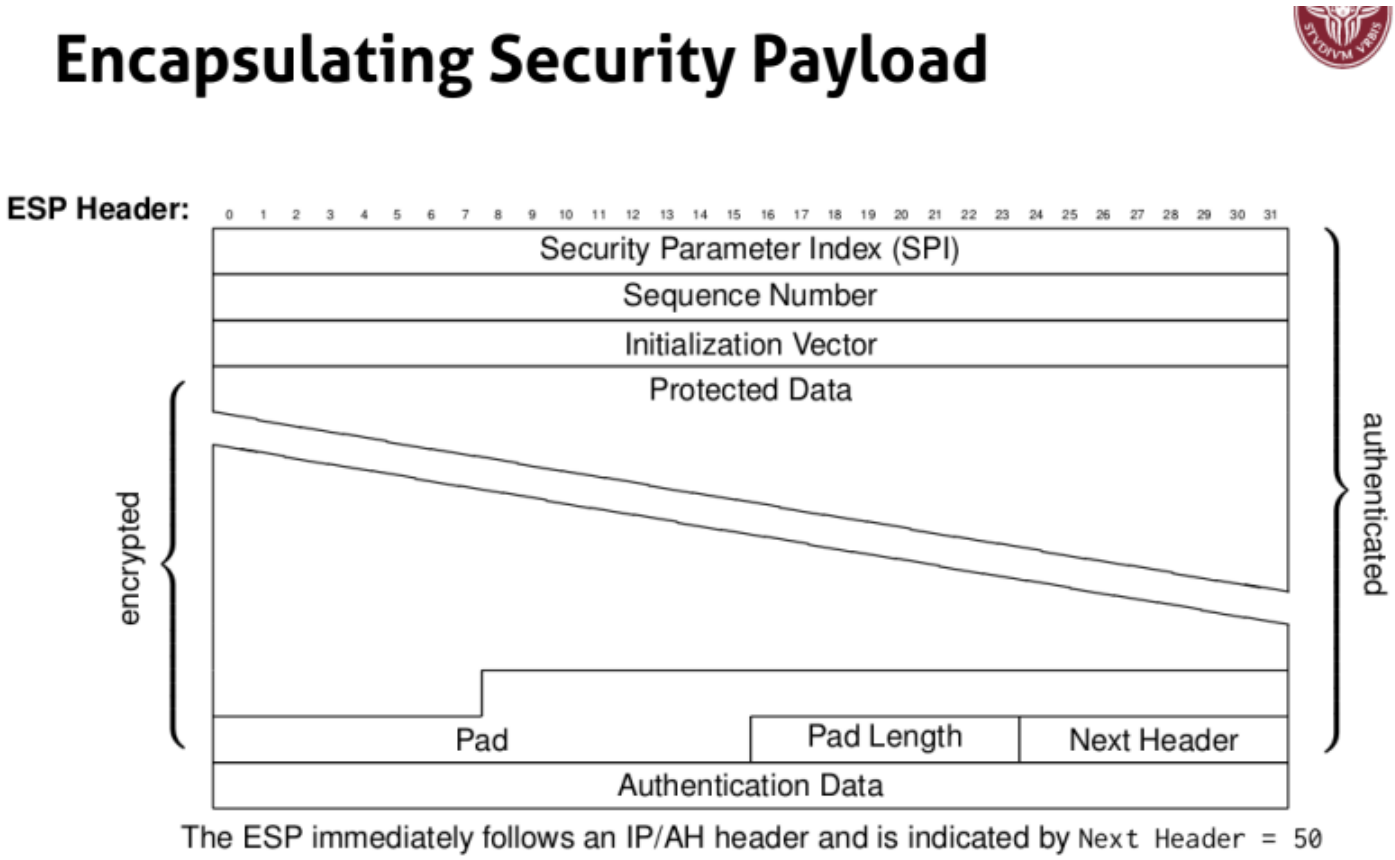


Figure 17: The ESP header

10.8.12 Authentication with IPv4 and IPv6

The authentication header is one of the possible IP header fields. It contains:

1. Next header: type of following header field
2. Payload length: length-2, in 32-bit words, of the AH
3. SPI: identifies the SA in use
4. Sequence number: monotonically increasing packet counter value
5. Authentication Data: a variable length HMAC based on MD5 or SHA-1 cryptographic hashing algorith, or AES-CBC, evaluated over: the immutable or predictable IP header fields, the rest of the AH header apart from this field, and all embedded payload (from t-layer or embedded IP packet), assumed immutable

Immutable fields do not change as the packet traverses the network (e.g., source address), meanwhile mutable but predictable fields may change, but can predicted (e.g., destination address). Mutable, unpredictable fields include: time-to-live, header checksum.

In IPv4 the AH header is inserted after the outermost IP header, depending on whether transport mode or tunnel mode is used.

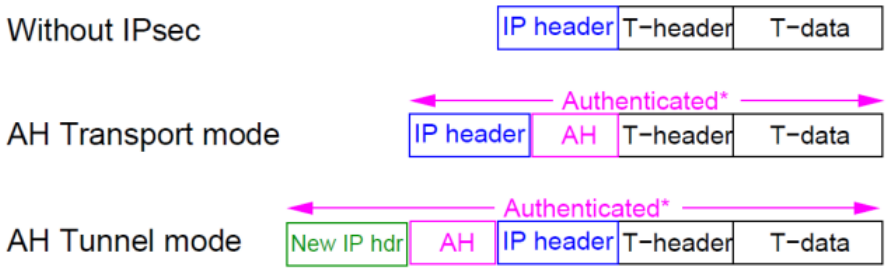


Figure 18: Authentication header in different modes - IPv4

In IPv6 the AH header is insterted after the outermost IP header, depending on whether transport or tunnel mode is used; notice that the integrity check (and thus authentication) does not cover any mutable, unpredictable header fields.

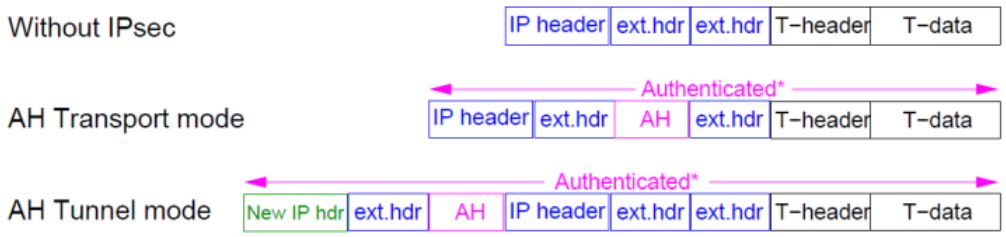


Figure 19: Authentication header in different modes - IPv6

10.8.13 ESP with IPv4 and IPv6

The ESP header is inserted after the outmost IP header, depending on whether transport or tunnel mode is used. Padding is added to the end of the transport layer payload to give traffic analysis protection; the ESP trailer and (optional) ESP authentication fields are added after the end of the padded transport layer payload. As usual, the integrity check (and thus authentication) does not cover any mutable unpredictable header fields.

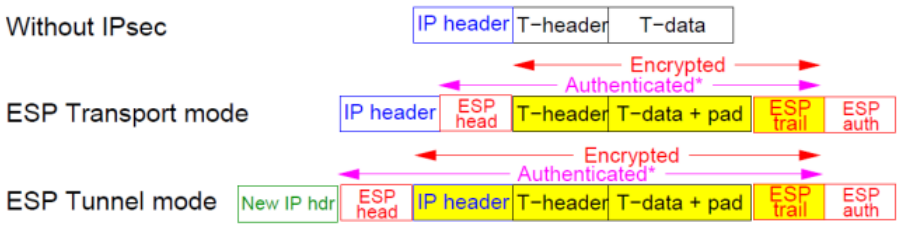


Figure 20: ESP with IPv4

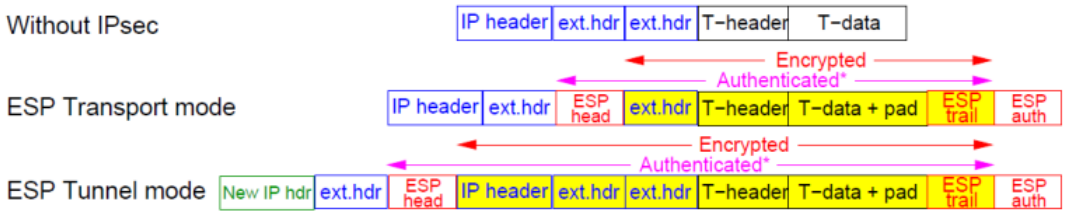


Figure 21: ESP with IPv6

10.8.14 Encryption + Authentication

It can by achieved by:

1. ESP with authentication; firstly ESP is applied to data, and then the AH field is added. In this there are two subcases: Transport mode: E+A applied to IP payload, but the IP header is not protected; Tunnel mode: E+A applied to the entire inner packet
2. Transport adjacency: uses bundled SAs, first ESP, then AH
3. Encryption covers original IP payload. Authentication covers ESP + original IP header, including source and destination IP addresses
4. Transport-Tunnel bundle. Used to achieve authentication before encryption, for example via inner AH transport SA and outer ESP tunnel SA
5. Authentication covers IP payload + IP immutable header. Encryption is applied to entire authenticated inner packet

10.8.15 Internet Key Exchange protocol v2 (IKEv2)

It's a standardized authentication and key management protocol to dynamically establish SAs between two endpoints. IKEv2 provides unified authentication and key establishment (IKEv1 was poorly specified), and tries to achieve trade-off between features, complexity and security.

Its main features are: it runs on UDP ports 500 and 4500, support for DoS mitigation hrough cookies, integrated support for requesting an IP address (useful for VPNs), negotiation of cryptographic suites (a complete set of algorithms used for SAs), and mutual authentication of the initiator and responder.

IKEv2 communication consists of message pairs, request and response; one pair is called an exchange, and an IKEv2 protocol run starts with two exchanges: IKE_SA_INIT and IKE_AUTH.

IKE_SA_INIT negotiates: encryption algorithm, integrity protection algorithm, diffie-hellman group, and pseudo random function prf.

IKE_AUTH realizes authentication via public key signatures or long-term pre-shared secret; a signed a block of data (AUTH) is transmitted through the IKE_AUTH exchange, and then authenticated by verifying the validity of the received AUTH payload.

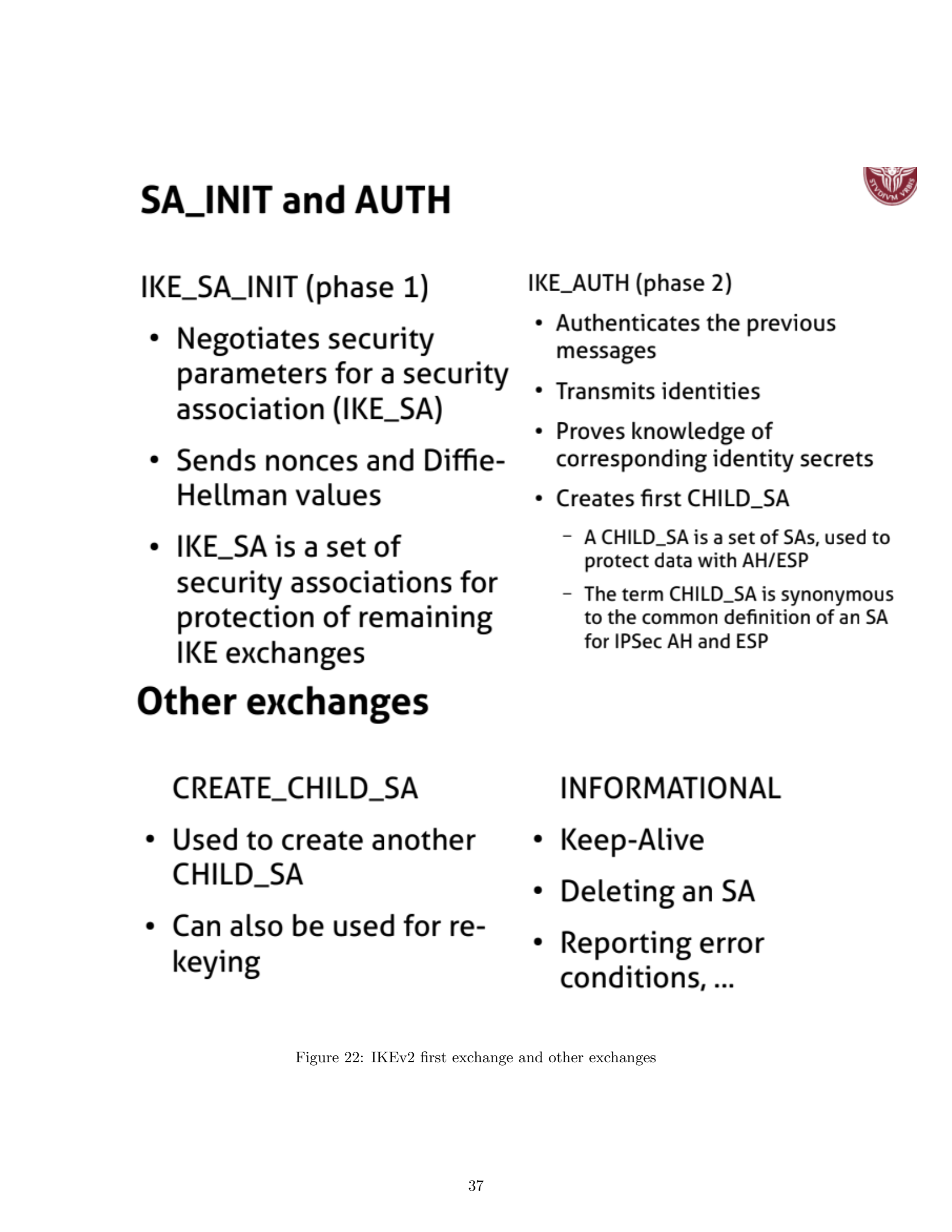
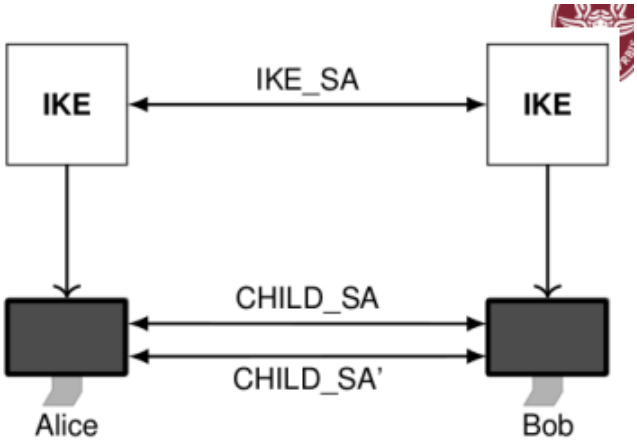


Figure 22: IKEv2 first exchange and other exchanges

IKE outcome



- IKE_SA is a set of Security Associations established after the initial IKEv2 exchange (IKE_SA_INIT)
- IKE_SA is used to encrypt and integrity protect all the remaining IKEv2 exchanges
- CHILD_SA is a set of Security Associations used to protect IP traffic with the AH/ESP protocol
  - AH provides data integrity and replay protection
  - ESP provides data integrity, replay protection and encryption

Figure 23: IKE_SA details

10.9 IPsec vs TLS

TLS is much more flexible because it's in an upper layer; it also provides apprication end-to-end security, best for web application (HTTPS). IPsec must run in kernel space, and is much more complex and complicated to manage with.

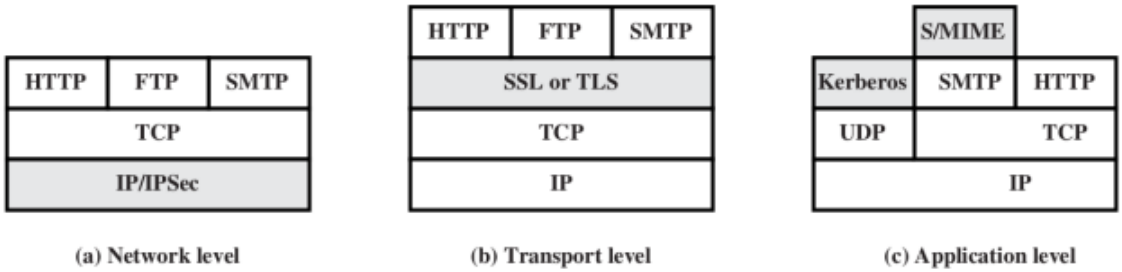


Figure 24: IPSec and TLS architectural difference

11 Proxy

11.1 Forward proxy

A forward proxy basically handles the request of a client with the target server. It provides: authentication, authorization, auditing (analyze data and provides info about the security), whitelisting, blacklisting, and caching.
With HTTP the proxy will be the middle point, forwarding the request toward the final termination, and reporting back to the source with HTTP. For other protocols it's possible to do HTTP tunneling. Notice that for example HTTPS can be done withouth tunneling in this way: the proxy will authenticate to the server and communicate via HTTPS with it, meanwhile its communications with the client will be in plain HTTP; this is obviously not good. With HTTP tunneling, the proxy establishes the TCP connection and becomes the middle-point, streaming the unmodified TCP data to and from the client; the proxy can still perform authentication, whitelisting and so on before accepting to forward the stream of data.

11.1.1 HTTP connect

Full specification here.
The HTTP tunnel is implemented as follows: the client connects to the proxy server, and uses the CONNECT method to specify the host name and the port number to connect to, followed by a string specifying the HTTP version, and some other HTTP request header if needed. The proxy then establish the connection with the server, and the tunneled data transfer can begin.
Notice that anything that supports a two-way TCP connection can be passed through a CONNECT tunnel (e.g. SSL/TLS, HTTPS).

11.2 Content filtering proxy

After authentication, HTTP proxy controls over the content that may be relayed, filtering websites, file types and so on.

11.3 Anonymizer proxy

It's a proxy server that acts as an intermediary and privacy shield between a client and the rest of the internet, accessing this latter on his behalf, protecting personal information (e.g. the IP address) by hiding them. The server sees the request coming from the proxy address rather than the actual client's address.

11.4 SSL forward proxy

It's a way to decrypt and inspect SSL/TLS traffic from internal users to the web, generally implemented in firewalls.
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 that is destined to the client, it decrypts it into clear and applies security profiles; then, it re-encrypt and forwards it to the client.

11.5 Reverse proxy

Reverse proxies operate on behalf of the server; it receives the requests from outside and then forward them to the actual destination. His main functions are: load balancing, cache static content, compression, accessing several servers into the same URL space, securing of internal servers, application level controls, and TLS acceleration.
No direct connection to the server means that reverse proxies also shield against DoS attacks.
They also provied HTTPS conenction to servers that support only HTTP, and can add AAA services (authentication, authorization, accounting) to server that do not have them (like IoT devices).

Reverse proxies can also be used for application control; they are called application firewalls, and operate at the application layer. A WAF (web application firewall) inspects HTTP traffic and prevents attacks such as SQL injection, cross-site scripting, file inclusion, and other type of security issues; an example of WAF is ModSecurity. They can detect whether an unwanted protocol is being provided through a non-standard port, or if a standard protocol is being abused in any harmful way.

TLS acceleration consist on speeding-up the time required by the SSL/TLS handshake process, who impact negatively on performance. Possible solutions are: SSL acceleration, to use hardware support to perform modular operations with large operands, and SSL offloading, to use a dedicated server only for SSL handshakes.
To perform SSL offloading, the proxy can decrypt the TLS/SSL-encrypted data and send it to the server in an unencrypted state; this also allows IDS or application firewall inspection, and it's called SSL termination. The proxy can also proxy intercepts and decrypt TLS/SSL-encrypted traffic, examine the content to ensure that it doesn't contain malicious code, and then re-encrypt it and send it to the server; this allows inspection of traffic but only before it reaches the server, and prevents applciation-layer attacks. This is called SSL forwarding.

11.6 Proxy and HTTPS

If a proxy wants to be able to read HTTPS traffic (notice that until now we were able to stream encrypted traffic unmodified), then it must perform a man-in-the-middle attack, pretending to be the server and the termination of the SSL/TLS connection. A possible solution is to use SSL bump: use the requested hostname to dynamically generate a server certificate and then impersonate the named server. What if a webserver hosts several websites? Each website has to have its own certificate; if HTTPS is used, then the SSL/TLS connection requires a certificate to be sent by the server, but if the hostname is encrypted (because we use HTTPS), then how to know which certificate must be sent?

11.6.1 Server Name Indication (SNI)

It's an extension to TLS by which a client indicates which hostname he's attempting to connect to at the start of the handshake process, so it is in clear text. This allows a server to present multiple certificates on the same IP address and TCP port number, allowing multiple secure websites that use HTTPS (or other security service over TLS) to be served by the same IP address.

With this protocol an eavasedropper can see which hostname is being requested, and governments can filter out some hostnames.

11.7 SOCKS proxy

It's a circuit level gateway (works at the session layer) that is similar to HTTP CONNECT proxy, but more versatile. It provides many authentication mechanisms, can tunnel TCP, UDP and IPv6, and can also work as reverse proxy. It's implemented in SSH, putty and Tor.

11.8 Transparent proxy

It's a proxy that appears as a packet filter to users, and a standard proxy to servers.

It works like this: at the start of a session, a TCP packet with a source address of the client and a destination address of a server travels to the proxy system, expecting to cross it like a normal IP gateway; the proxy's TCP/IP software stack sees this incoming packet for a destination that is not one of its own address. Instead of forwarding or dropping the packet, it accepts it, pretending to be the server; then, it operates like a classic proxy, as a middle point between the client and the server.

The difference with a standard proxy is that in a standard proxy the user configurates its browser (or similar), and consciously send packet to the proxy, meanwhile in a transparent one the user doesn't configure anything and thinks to send packet to the server, meanwhile the proxy intercepts it.

A problem that will be faced while using HTTPS (or other TLS basedprotocols) is that the communication between the proxy and the server will be done using the server certificates, meanwhile between the proxy and the client, the proxy's certificate will be used. The proxy then can't store the access log since the destination URL is unknown.

Transparent proxies intercepts the HTTP requests that have the IP address of the server, and search for the hostname within this request (for HTTPS requests see the secion on SSL bump). How to interecpt HTTP/HTTPS requests?

11.8.1 Policy Based Routing

In traditional routing, all the routing is a destination driven process. What about making routing decisions based on other information?
To intercept desired traffic we can use policies, e.g. any TCP connection with port 80 must be forwarded to the transparent proxy. Then, we will need a new routing table with a different next hop: the proxy. This can be done with iptables and the "packet marking".

11.8.2 ICAP

The Internet Content Adaption Protocol is a lighweight protocol for the execution of remote procedure call on HTTP messages; it allows ICAP clients to pass HTTP messages to ICAP servers for some sort of transformation or other processing; the messages will then sent back to the clients, with the applied modifications. Some examples of usage are: translation of the message, scan for viruses, and content filtering. Such tasks can be performed by the origin server too, but maybe it's preferred to perform them wih an ICAP server. ICAP can be coupled with a transparent proxy to transparently provide additional functionalities, using a standardized interface towards ICAP servers.

12 Intrusion Detection Systems

An IDS aims at detecting the presence of intruders before serious damage is done. The ultimate purpose of an intruder could be: prevent the legitimate users from using the system, reveal confidential information, and use the system as stepping stone to attack other systems.

Second generation IDS are IPS, Intrusion Prevention Systems, who also produce **responses** to suspicious activity, for example blocking switches ports or adding new firewall rules.

In particular, IPSs can: drop packet, terminate sessions, modify firewall rules to block suspicious hosts, traffic shaping for slowing down less critical traffic (such as P2P, videos), and generate alerts and logs.

12.1 IDS vs IPS

Both of them perform deep packet inspection (payload), and the IDS report intrusions by out of band detection, meanwhile the IPS blocks intrusions by in band filtering.
Out of band means that the IDS connected to both the external network and the internal network, but doesn't reside inside of them; in line or in band means that the IPS is on the path between our internal network and the external network.

IDS works passively, meaning that it detect and raise alarms, meanwhile the IPS is active, meaning that it detects and reacts.
Alarms can be true positive, true negative, false positive and false negative. IDS must raise few false positive, but still be able to report as many true positive as possible.

An IPS must be such that no false positives can occur, otherwise legitimate traffic will be blocked; IDSs on other hands will not interfere with traffic.
Furthermore, to match the line speed, IPS hardware requirement is higher.

12.2 other functionalities

Other functionalities performed by IDS/IPS are: recording information related to observed events, alert securityadministrator of important observed events, producing reports (who summarize the monitored events or provide details on particular events of interest), and in case of IPS, changing the network activity: drop connections, block accesses, change of configurations of other devices, change of content of packets, and so on.

12.3 Activities monitored by IDS/IPS

They monitor: every activity that is sensible to occurrences of certain events, pattern of specific commands in application sessions (e.g. login and location frequency), content types with different fields of application protocols (e.g. the password field must be in the right format to avoid SQL injection), network packet patterns between protected servers and the outside world (client application, protocol and port, volume and duration but also rate and burst length distributions for traffic to prevent DoS attacks), and privilege escalation. They also monitor attacks by legitimate users/insiders, such as: illegitimate use of root privileges, unauthorized access to resources and data, command and program executions. And of course they monitor also malwares: rootkits, trojans, spywares, viruses and scripts.

12.4 Typs of IDS

1. Host-based (HIDS): Monitors events in a single host and is typically deployed on critical hosts offering public services. The advantage is that there's a better visibility into the behaviour of individual applications running on the host.

2. Network-based (NIDS): analyses network, transport and application protocol activity; it's often placed behind a router or firewall that is the entrance of a critical asset. The advantage is that a single NIDS/IPS can protect many hosts and detect global patterns.

3. Wireless (WIDS): analyses wireless networking protocol activity (but no transport/application level); it's tipically deployed in or near an organization's wireless network.

12.4.1 HIDS

HIDS monitor traffic only on one specific system, so it doesn't use the promiscous mode.
It looks out for unusual events or patterns that may indicate problems, such as unauthorized access and activity, unexpected activity, changes in configuration, and software changes.
Their advantages are:

1. They protect mobile hosts from attack when attached outside the protected network

2. They protect against local attacks from a user/removable devices

3. They protect against attacks from the same subnet/LAN

4. They protect against encrypted attacks where the encrypted data stream terminatesat the host being attacked

5. They can inspect packet content after decrypting received VPN or SSL packets

6. They can inspect packet together with an anti-malware software

Their cons are: if an attacker takes over a host, then IDS can be tampered and audit logs can be modified; the attack is seen only from the perspective of a host, and lastly this type of detection has a high false alarm rate.

12.4.2 NIDS

It usually work in promiscous mode, and can have multiple NICs to monitor multiple network segments.
It's usually connected to switches with ports mirrored (or in SPAN mode), so all the traffic generated within all the ports of the switches are replicated on the mirrored port where the NIDS is placed.
Often it has a series of sensors placed in different networks (is a distributed detection system).

They monitor and protect against network oriented attacks (DDoS, bandiwdth consumption), and are independant of the host's OS. They protect network equipment who cant/don't have HIDS, such as printers, and monitor user activities looking into data portion of the packets for mailious command sequences.

However, they might be unable to: detect encrypted traffic, detect some attacks in the hosts, and they require high computation capabilities.

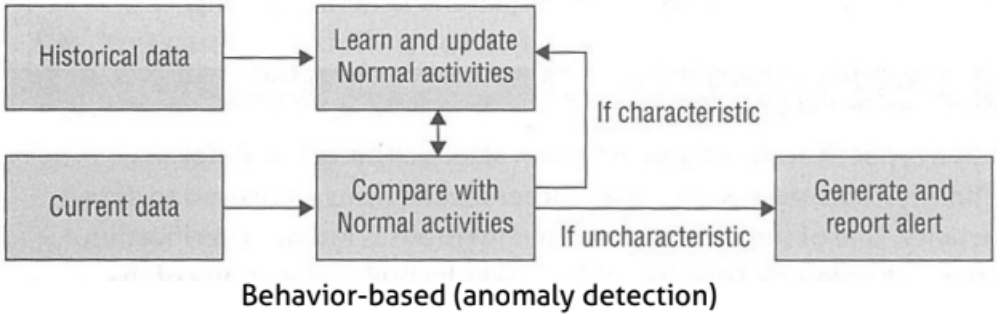
12.4.3 NIDS and HIDS combined

There are attacks that can be detected only by NIDS (e.g. routing advertisement injection), and attacks that can be detected only by HIDs (e.g. local privilege escalation), so integrating both of them result in a more accurate filter/block/quarantine of infected mediums.

12.4.4 Other types of IDS

Other types of IDS are: file integrity monitors (monitor changes to key system configuration files), flow-based IDS (netflow), and hybrid detection capabilities. Flow-based IDS tracks network connections, establishes patetrn of normal traffic, and alerts when unusual services/patterns/behaviours/protocols are seen; it's good for big networks. Hybrid detection systems augment or replace signature-based detection; they are usually behaviour-based (behaviour is analyzed by a pseudo artificial intelligence to learn what is normal and what is not), and need training periods to establish a baseline.

IDS approaches



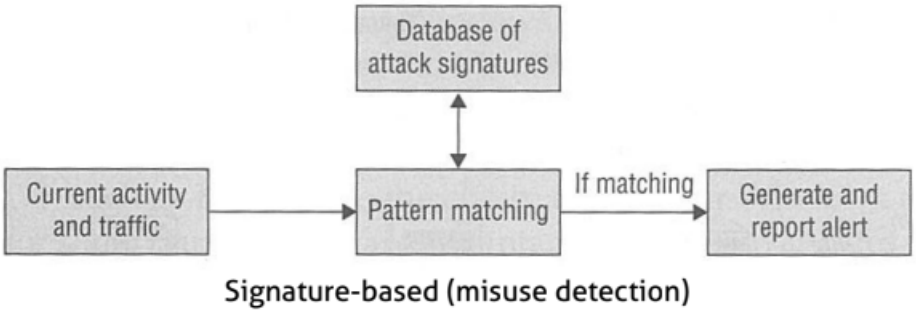


Figure 25: Different types of IDS

12.5 How to recognize an intrusion

There are two ways: behaviour based and signature based. The behaviour based detect anomalies, and does so: defining behavioural characteristics of normal behaviour, then compares actual behaviour with these; if there's a significant difference, then it raise an alarm. It's difficult to define all the possible normal behaviours; new activities often give false positives.
Signature based systems detect misuse, and do so: defining characteristics of various types of abnormal activities, then compares actual behaviour with these, and if any of hem match, it raise an alarm; it's difficult to produce a complete catalog of abnormal activities, and if any is missing, then there will be false negatives.

Different technologies need their own feature sets, e.g. wireless networks have the following features: signal power, sequence number "jumps", and round-trip time.

12.5.1 Behaviour based

Behaviour is typically described in terms of a set of features that should be able to describe all the relevant aspects of the behaviour to be recognized. Anomaly detection needs some sort of learning, and it's usually based on data mining actual observations. A feature set that is too large means that both training and classification will take unnecessarily long time.

The questions are, how do we recognize normal and abnormal behaviour? How quickly? and how to deal with abnormal systems?
The process is: take sequence of observed behavioural elements, like system calls, network packets and so on; then derive feature values from them, and then derive the normal behaviour using either statistics, a set of rules, or a machine learning approach. Then we can compare the traffic against the normal behaviour, using distance metrics for the statistical approach, probability measures (how likely is this sequence?) in the case of machine learning, and a rule set in the last case.

Adaptive profiles can account for normal network changes to avoid raising false alarms; self-learning is critical to ensure wide and successful deployment of anomaly-based detection mechanisms (manual setting of profiles is too difficult because a network is really complex).

Notice that **anomaly based systems can not be used for IPS**, because they lead to false positives.

Protocol anomaly for the purpose of dentifiying a protocol anomaly, the layers 2-7 are inspected, and a suspicious protocol or service is identified if its purpose or its ports are not standard, e.g. modified protocols for tunneling through firewalls, like P2P on port 80. Full IP defragmentation (fragmented packet are reassembled) and TCP reassembly is performed, with a deep packet inspection, looking for:

1. IP fragmentation overlap
2. Suspicious IP options
3. Unusual TCP segmentation overlap
4. Illegal TCP options and usage

A deep application protocol parsing and deconding is also performed, looking for:

1. Illegal field values and combinations
2. Illegal command usage
3. Unusually long or short field lengths, (maybe) indicating buffer overflow
4. Very long arguments to string functions
5. Unusual number of occurrences of particular fields/commands

The application semantics is checked, checking that: type of encoding is legal for a given field, no application is embedded within, and no shellcode is in unexpected fields.

Statistical anomaly Anomaly can also be detected using statistical measures; they are used to capture network traffic behaviour. The balance among different types of TCP packets (in the absence of attacks) can be learned and compared against short term observations that will be affected by attack events. Profiles based on statistical measures time-of-the-day, day-of-the-week variants in the traffic volume. Profiles for traffic rate distributions on a multi-week scale normal network environments. All of these profiles may raise DDoS anomalies based on rare events, such as a long bursts of high-rate traffic, and on differences in the long and short term distributions.

12.5.2 Signature based

Signature base systems starts from the idea that intruders may have a characteristic appearance which makes possible to identify them. So payloads of packets are screened looking for specific patterns (signatures). Suppliers of IDSs maintain a huge database of signatures which characterize various classes of intruder, so signatures are typically supplied by the manufacturer of IDS.
Rapid recognition involves searching for matches for one or more of the known signatures from a collection of many thousand signatures.
Rules express actions on given conditions, possibly with complex predicates, including timing, payload, content etc. Rules can be derived from automatic IDS technologies (markov, ANN clustering, etc.), but they're usually made by experts.

```
Act.  proto src...     dest...                  options...
alert tcp any any -> 10.1.1.0/24 80 (content: "/cgi-bin/phf";)
alert tcp any any -> 10.1.1.0/24 6000:6010 (msg:"X traffic";)
alert tcp any any -> any 21 (msg:"FTP ROOT";content:"USER root";)
```

Figure 26: Example of IDS signature based rules

Signature based IDSs are packet sniffer on steroids; they capture packets in a LAN and apply some complex logic to decide whether an intrusion has taken place. An example of signature based IDS is SNORT.
The problem is that they can't inspect encrypted traffic (VPNs, SSL), they must process a huge amount of packets, and of course not every attack comes from the network.

IDS decode packets (performs deep packet inspection), and decode application and protocol headers to look at high layer activity.

The set of rule define a behavioural signature likely to be associated with an attack of a certain type; attack signatures are usually very specific and may miss variants of known attack. An example of suspicious behaviour: a lot of SYN packet without ACKs coming back; it could be a DoS attack, or a poor network connection.

But how do we extract misuse signatures? Honeypots are used for this task: they are an easy target for attacks, and they're used to attract attackers. An honeypot is usually a computer, meanwhile an honeynet is a network of computers.

But it's easy to evade IDS rules. For example: we want to detect the string "USER root" in a packet stream; scanning for every packet is not enough, because the string could be divided in several packets, and a stateless NIDS can't identify it. Recording previous packet's text is not enough, because the string may be sent out of order; a full reassembly of TCP state is not enough, because there are a number of TCP tricks to make sure that the NIDS see certain packets, but then they are dropped by the receiving application. For example, the string might be sent with a random packet in between so the NIDS can't identify the attack, but then the host will drop it (maybe because it has a wrong checksum, or because it has a shorter TTL) and successively read the incriminated sequence.

12.6 Signature vs Behaviour

Signature based detection can clearly indicate the detected attack method; the behavioural based alerts can indicate: the attack type, the behavioural rule that was violated (such as port scan), or the statistical profile that was violated. In general, the behaviour based system can't identify the specific attack or the exploit that was detected, but it requires a security administrator to investigate it given the clues of the behaviour rule. This is acceptable for new attacks, but established and known attack should be easily identifiable. Deciphering informaiton about each attack detected by a behavoural system is unmanageable for a large number of hosts.

12.7 Combine behaviour and signature

If used combined, then: if a false negative occurs, the problem is in the signature based misuse detection; if there's a false positive, then harmless behaviour is clasified as an attack, and the problem is in the anomaly detection. If combined, the two systems can virtually remove false positives and false negatives. The system will be able then to avoid DoS/DDoS attacks and zero-day exploit without updating the system, and to perform interdependance and cross-checking of suspicious traffic. Once an exploit has been recognized with the behaviour system, then a stateful signature can be created to provide future accurate detection and save manpower.

12.8 Distributed IDS

The goal is to extend the focus from single systems to information infrastructure, to monitor and correlate public, internal VLANs, and DMZ segments of the IDS/IPS sensors and firewalls. Correlation among these segments yield an accurate picture of the network attacks that were either blocked or made it into the internal network.

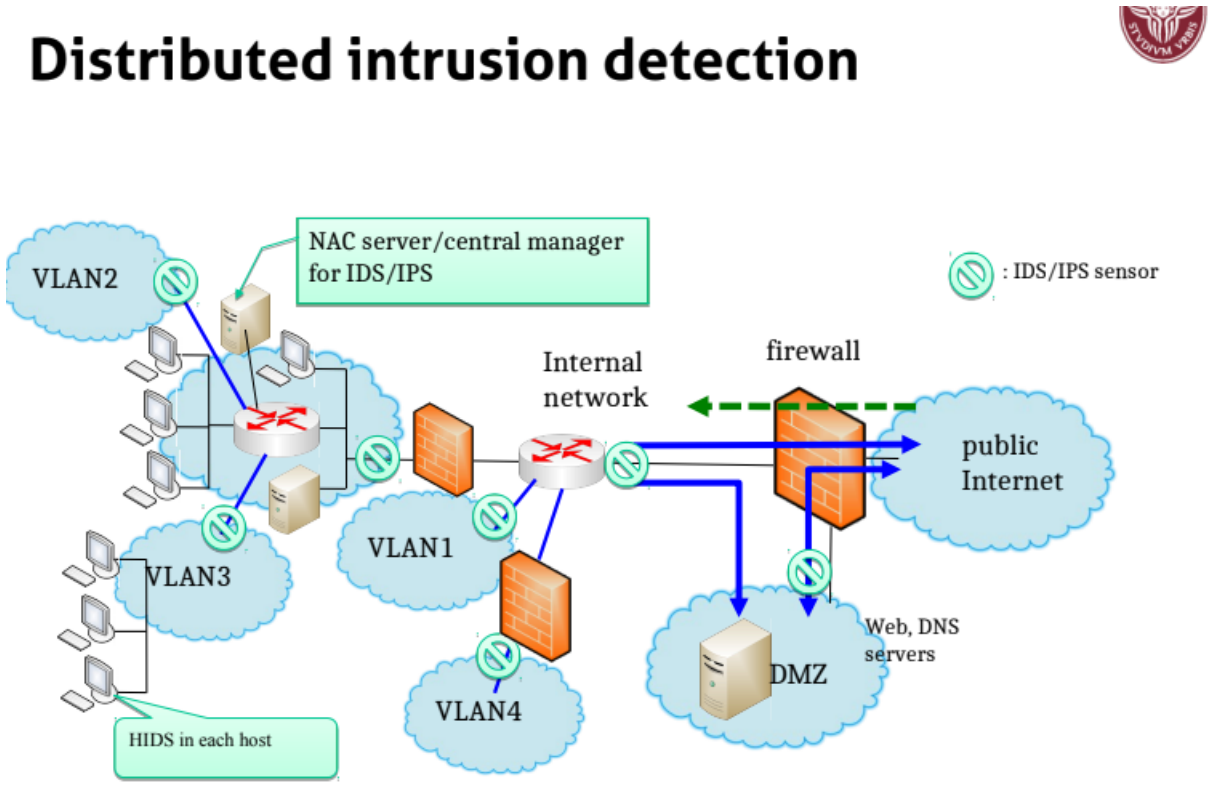


Figure 27: Distributed IDS architecture

12.9 State information and analysis

State information is captured and updated in real time; maintaining state information enables sensorto gain context for attack detection. It is the basis for layer 2-7 detection, and helps to detect malwares, trojans, key loggers, botnets and worms. It is useful to utilize multiple token matches to capture attack signatures/behaviours that span packet boundaries or are out of order in a stream.

12.10 Normalization

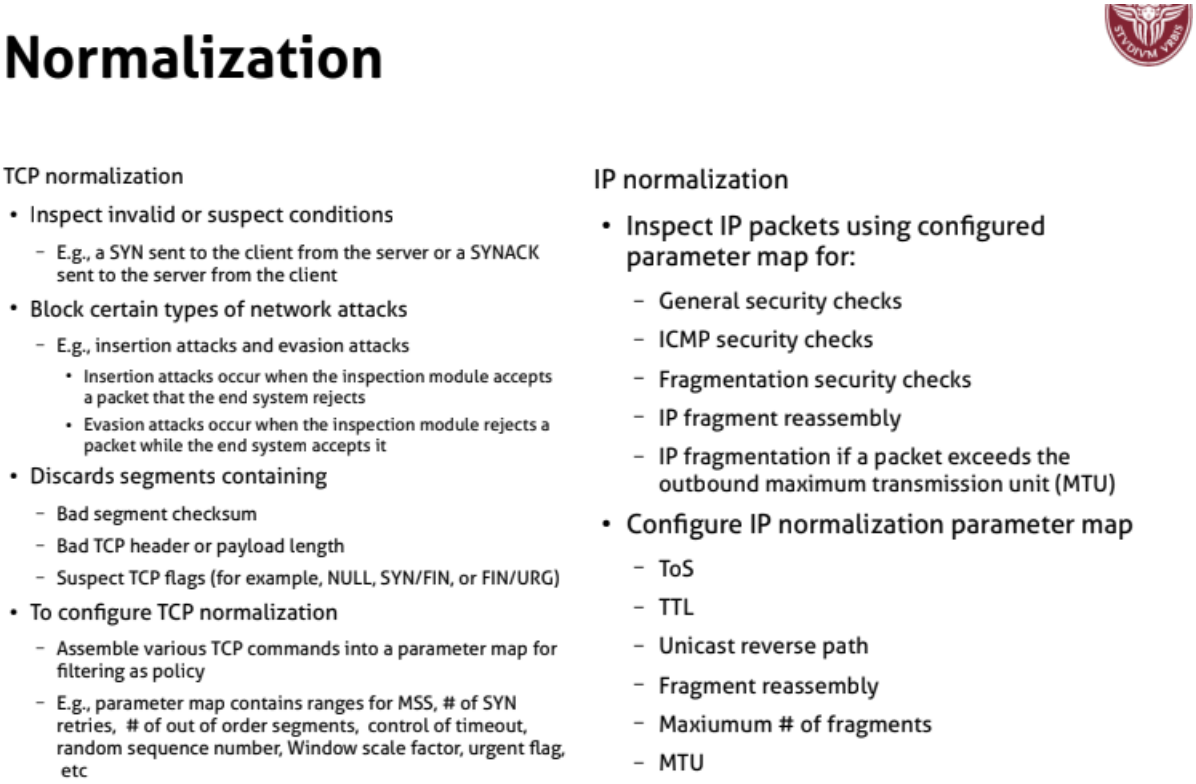


Figure 28: IP and TCp normalization

12.11 SNORT

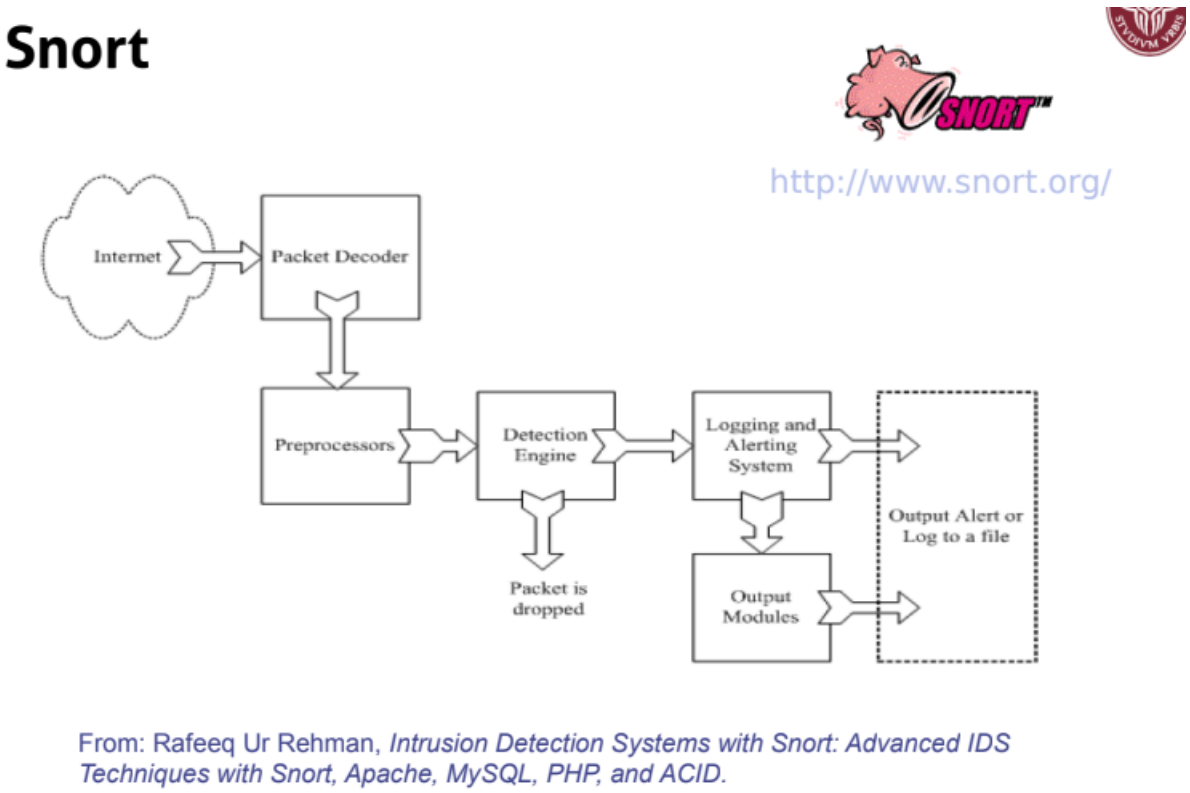


Figure 29: SNORT architecture

SNORT is an open source NIDS/NIPS, and is the most deployed IDS worldwide. It combines the benefits of signature, protocol and anomaly based inspection methods.
Its main components are: packet decoder (input from ethernet, PPP, etc.), preprocessors (detect anomalies in packet headers, decode HTTP URI, perform packet defragmentation and reassembly of TCP), detection enginge (applies rules to packets) logging and alerting system, and output modules (alerts, log or other outputs).

12.11.1 SNORT rules

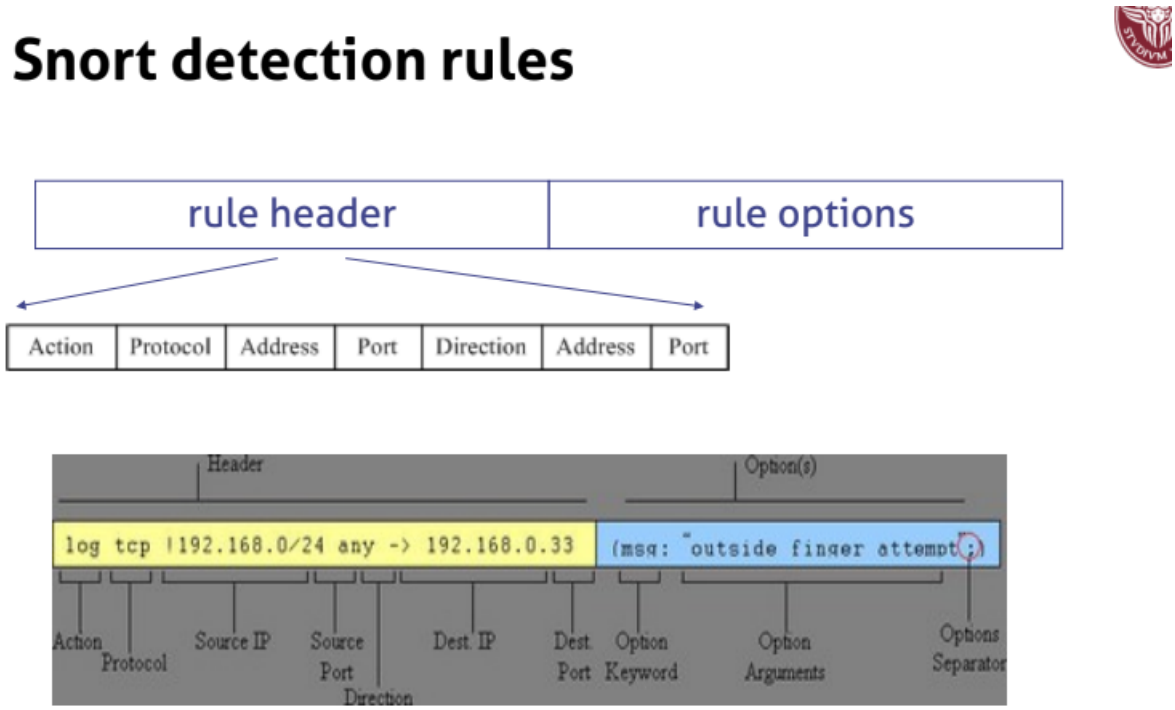
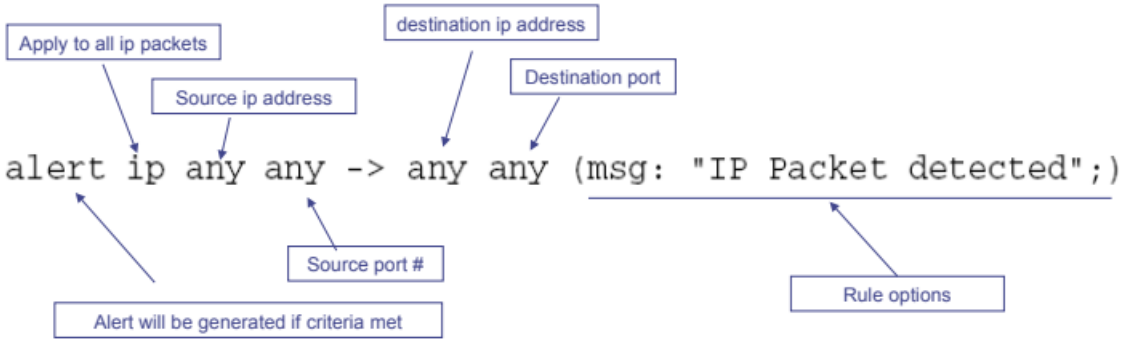


Figure 30: SNORT rules

Example




```
alert tcp $TELNET_SERVERS 23 -> $EXTERNAL_NET any (msg:"TELNET
  Attempted SU from wrong group"; flow:
from_server,established; content:"to su root"; nocase;
  classtype:attempted-admin; sid:715; rev:6;)
```

Figure 31: An example of a SNORT rule

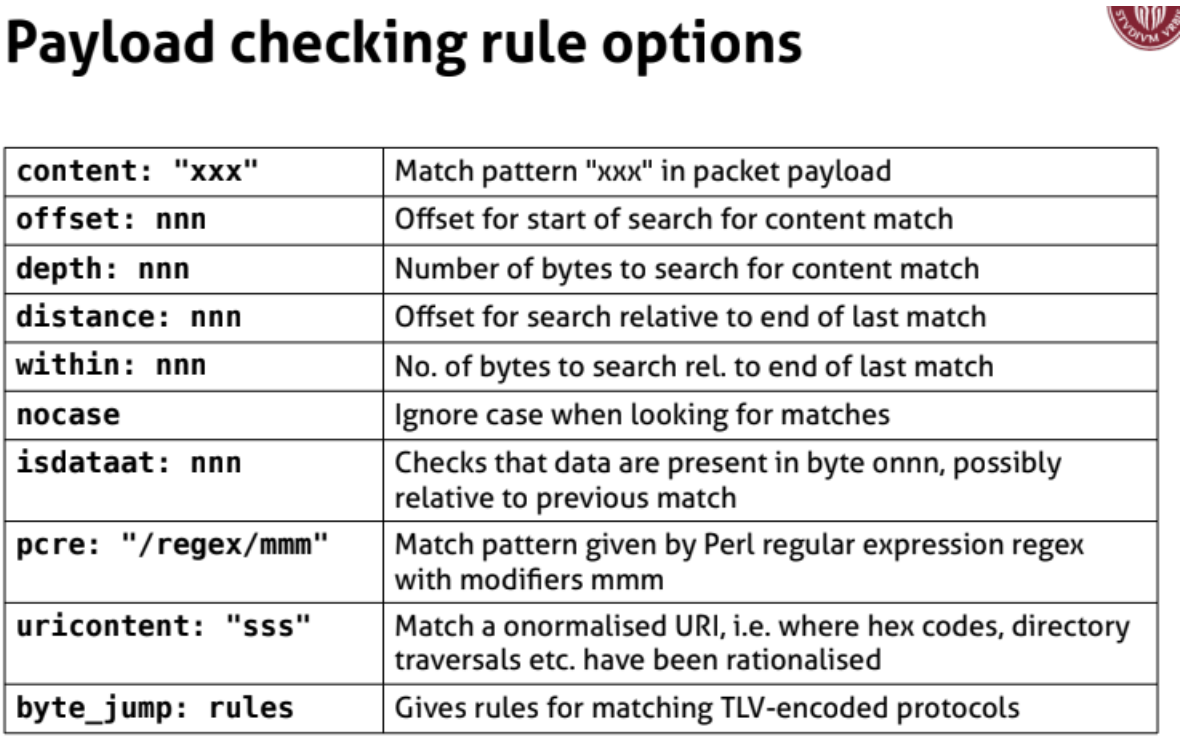
12.11.2 SNORT rules for headers and payload

TCP/IP header rule options

ipopts: opt	Match specific IP option opt
flags: fff	Match settings of one or more TCP flags
seq: nnn	Match specific TCP sequence number
ack: nnn	Match specific TCP ack number
window: nnn	Match specific TCP window size
itype: nnn	Match ICMP type field value (or range)
icode: nnn	Match ICMP code field value (or range)
fragbits: mmm	Match IP fragmentation/reserved header bits
ip_proto: proto	Match IP Protocol field by number or name
id: nnn	Match value of IP ID header field
ttl: nnn	Match value of IP TTL header field
dsize: nnn	Match packet payload size (or size range)
flow: flowstat	Match flow direction/state
rpc: app,ver,proc	Match RPC application, version and procedure

Figure 32: Rules on the TCP/IP header

12.11.3 Rule options



Payload checking rule options

content: "xxx"	Match pattern "xxx" in packet payload
offset: nnn	Offset for start of search for content match
depth: nnn	Number of bytes to search for content match
distance: nnn	Offset for search relative to end of last match
within: nnn	No. of bytes to search rel. to end of last match
nocase	Ignore case when looking for matches
isdataat: nnn	Checks that data are present in byte onnn, possibly relative to previous match
pcre: "/regex/mmm"	Match pattern given by Perl regular expression regex with modifiers mmm
uricontent: "sss"	Match a onormalised URI, i.e. where hex codes, directory traversals etc. have been rationalised
byte_jump: rules	Gives rules for matching TLV-encoded protocols

Figure 33: Rules for the payload

12.12 Suricata

It's a high performance NIDS,IPS and network security monitoring engine. It's open source, and has been developed to overcome SNORT limitations. It also provide offline pcap processing.

Suricata's workload is distributed thanks to multi-threading and GPU acceleration; it supports packet decoding of: IPv4, IPv6, TCP, UDP, SCTP, ICMPv4, ICMPv6, GRE, Ethernet, PPP, PPPoE, Raw, SLL, VLAN, QINQ, MPLS, ERSPAN.
It can decode application layer of the followings: HTTP, SSL, TLS, SMB, DCERPC, SMTP, FTP, SSH, DNS, Modbus, ENIP/CIP, DNP3, NFS, NTP, DHCP, TFTP, KRB5, IKEv2.

Suricata can be integrated with other solutions, suck as SIEMs and databases using YAML and JSON files as inputs and outputs.
Suricata also incorporate Lua scripting language to create rules that identify conditions that would be difficult or impossible to describe with a legacy SNORT rule.

12.13 Zeek (or BRO)

It's older than SNORT, nad has a c++ like language than a rules structure; it can be used as IDS but also as a network monitor.

12.14 fail2ban

It's an IPS framework that protects servers from brute-force attacks; it scans log files and bans IP addresses of hosts that have too many failures within a specified time windows. It's kinda like a dynamic firewall that detects incoming connection failures, and dynamically adds a firewall rule to block hosts after too many failures.
fail2ban: supports both clients and servers, is multi-threaded, autodetect datetime format, and has a lots of predefined support for services (like squid) and actions (like iptables).

It's limitations are the reaction time (it's a log parser, so it can't do anything before something is written on the log file), and the fact that a local user may initiate a DoS attack by forging the syslog entries with the logger(1) (the logger command create entries in the system log) command.

Terminology

- fail2ban
  - Software that bans & unbans IP addresses after a defined number of failures
- (un)ban
  - (Remove)/Add a firewall rule to (un)block an IP address
- jail
  - A jail is the definition of one fail2ban-server thread that watches one or more log file(s), using one filter and can perform one or more actions
- filter
  - Regular expression(s) applied to entries in the jail's log file(s) trying to find pattern matches identifying brute-force break-in attempts
- action
  - One or more commands executed when the outcome of the filter process is true AND the criteria in the fail2ban and jail configuration files are satisfied to perform a ban

Figure 34: fail2ban terminology

fail2ban components are: the content of /etc/fail2ban directory, the fail2ban-server (the core of the IPS), the fail2ban-client (the command-line-interface with the server), and fail2ban-regex (a command-line-interface to test regular expressions and filters).

13 SIEM

It's an approach to cybersecurity that combines SIM and SEM.
SIM stands for security information management, and collects log data for analysis, alerting responsible individuals of security threats and events.
SEM stands for security event management, and it conducts real-time system monitoring, notifies network admins of important issues, and establishes event correlations.

There is no standard SIEM protocol or established methodology, but most SIEM systems know how to automatically collect and process information from distributed sources, store it in one centralized location, correlate between different events, produce alerts and reports based on this information, and help for compliance and security incident management (digital forensics). They can be agent-based or agentless.
SIEM systems should provide the following set of services: log management, IT regulatory compliance, event correlation, active response, and endpoint security.

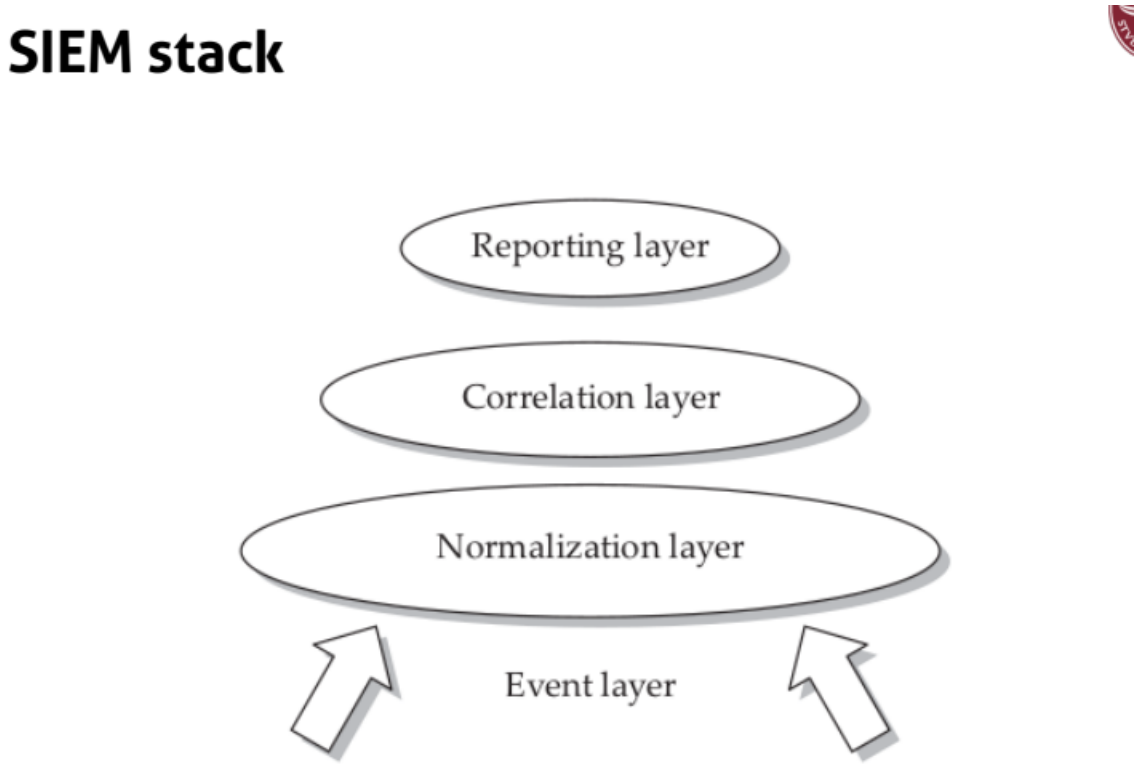


Figure 35: The SIEM stack

13.1 Log management

Logs are not events, but events can be logged.
Nodes in an IT system, and in particular the important ones, send relevant system and application logs to a centralized database that is managed by the SIEM application. This SIEM database application first parses and normalizes the data sent by the numerous and very different nodes in the IT system. To satisfy log management requirements, SIEM typically provides log storage, organization, retrieval and archival services.

The SIEM system makes use of near real-time analysis and data mining on the health and security status of all the IT systems feeding their data into the SIEM system; the more the nodes that feed into the SIEM system, the more complete and accurate the vision is of the IT system as a whole.

13.1.1 Log correlation

Log correlation means monitor the incoming logs for: logical sequences, patterns, relationships and values. The ultimate goal is to analyze and identify events that are invisible to individual systems; generally the system make use of supporting data.
These supporting data are data collected by other sources that can be imported into the SIEM to make comparative determinations; e.g.: geo-location information, names, IP addresses, operating systems, software versions, etc.. They can be used as weights to prioritize and escalate alerts.

Correlation engines are usually the most distinguishing feature of a SIEM; before searching for correlation, there's the need to perform normalization (logs are in standard but not homogeneous format, e.g. drop vs block). So, a common syntax for events must be defined to apply it to the log.
Then to find correlation there are rules that can trigger alerts or actions, or a machine learning approach can be employed.

13.2 IT regulatory compliance

Once logs are stored, filters or rules can be built to audit (monitor against a standard) and validate compliance, or to identify violations of compliance requirements imposed upon the organization.
Some examples: monitor the frequency of password change, identify OS or application patches that failed to install, audit the frequency of antivirus and IDS updates, etc..
SIEM generally can produce reports, often needed by business to provide evidence of self-auditing and to validate their level of compliance.

All forms of compliance ask the fundamental question related to diligence: Have you taken the steps to perform your responsibilities to securely manage the information in your control—which a reasonable person would expect of someone in your position?

For this reason evidences are fundamental, and the log server has to be reliable, using for example TCP and encryption; it could also be important to sign the logs, to provide authentication and integrity of them.

SIEM can also include compliance checklist (e.g. SPLUNK); the reports that SIEM can generate can be used as evidences for the IT regulatory compliance.

13.3 Event correlation

The correlation engine on a SIEM can investigate and consider (correlate) diferent 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, and can consider various conditions before triggering an alarm.

13.4 Active response

It's a SIEM mode that makes the SIEM trigger and generate an automated resposne to threat; the (active) response might be adding an IP and port filters on the ACL on a firewall.

13.5 Endpoint security

Most SIEM systems can monitor endpoint security to centrally validate the security health of a system; some SIEM can even manage endpoint security, actually making adjustments and improvements to the node's security on the remote system, e.g. configuring firewalls and updating/monitoring antivirus, antispyware and antispam.
Other actions that can be performed are: patching OSs and major applications, check configuration of firewalls, deployment HIDS and HIPS, management of removable devices, network access control, deployment of NIDS and NIPS.

13.6 Importance of logs

Logs are needed to extract information about the events that the network produces; without them, it's impossible to achieve any security management.
Typical questions are: how long do we retain (store) the logs? The system must find a balance between needs and desires; usually is agreed on the regulatory.
How much information is required to retain? What kind of information do we retain and analyze?

Which devices will the system collect events from? Critical servers, devices providing access to critical servers, IDS; optionally network endpoints.
Which events will be collected? Debug info, log-in records, configuration changes, alerts.
Where will we store the logs? On the local storage, cloud or hybrid solutions.

13.6.1 Log sources

1. Syslog of servers and end-user computers
2. Alerts from IDS/IPS and antivirus
3. Flow data
4. Domain controllers
5. Databases
6. Switches and routers
7. VPN gateways
8. Firewalls
9. Web filters and proxies

13.7 Additional features

SIEM: support for open source threat intelligence feeds, can perform real-time analysis and alert (IDS-like), can perform automated response (this is optional), has advanced search capabilities, and can do historical and forensic analysis.

13.7.1 Threat intelligence

Threat intelligence is the analysis of data using tools and techniques to generate meaningful information about existing or emerging threats targeting the organization;feeds and reports generated can help security officers in making decisions concerning organizational security.
Threat intelligence also helps organizations to make faster, more informed security decisions and change their behaviour from reactive to proactive to combat attacks.

13.8 Digital forensics

It's a branch of forensic science focused on the recovery and investigation of material found in digital devices and cybercrimes; it is concerned with the identification, preservation, examination and analysis of digital evidence, using scientifically accepted and validated processes, to use them in and out a court of law.

13.9 SIEM operational interfaces

It uses dashboards and maps, to present information in a way that administrators can understand at first glance, allowing them to see patterns, understand trends and identify unusual activities. Such dashboards and maps are a graphical and organized representation of alterts, event data and statistical information; they are also key differentiator among the different SIEM products on the market.

14 Penetration testing

It's a practice adopted by every company; a third part tries to violate the system to discover and exploit (without damage) vulnearabilities. Different methods of penetration testing roughly follow the same step, but they are adapted depending on the system; these steps are:

1. Planning
2. Intelligence gathering
3. Intrusive search
4. Threat modeling and vulnerability scan
5. Data analysis
6. Exploitation
7. Report

14.1 Planning

This phase is used to specify the part of the system that must be analyzed (providing parameters such as the IP address, domains, and SW that must be executed), and how the tester must operate, what he can and can't do, and in which hours/days.

14.2 Intelligence gathering

In this phase the company will gather as much information as possible to understand the target system; this can be done by searching on public sources (this is called reconnaissance), or other tools like nslookup, traceroute, etc.

14.3 Intrusive target search

This search probe and explore the target network (it has permission to do so, otherwise this step is illegal); it aims to: find live systems, discover open ports, services and enumeration. All of this can be done with a network scanner.

14.4 Threat modeling and vulnerability scan

Once the information are gathered and the company has a view of the system, it's time to plan consider the possible targets (obtain permissions, changing roles, etc.) and strategies. To examine information and find possible targets there are different authomatic tools, but the most interesting part is done manually by testers.

Usually, this phase is done by automatically scanning DB with known vulnerabilities, and check

whether they are present in the system. This pahse can be carried out inside or outside the network.

14.5 Data analysis

It's the step that tries to understand the data gathered with the previous steps; it determines missing elements or impartial results. This step can be repeated after the exploitation and can be very time consuming.

14.6 Exploitation

It's the process of validating a discovered vulnerability (it fails if the vulnerability cannot be exploited). There are tools to perform this phase, such as: metasploit, BeEF, Throwback.

14.7 Report

In this phase the company will report the steps that have led to vulnerability exploit, describing activities, suggesting best practices and mitigation techniques.
It will include: description of the exploited/exposed vulnerabilities, analysis of the finding extracted from the reports, recommendations of patches, best practices and solutions to mitigate or remove the risk.

14.8 Methodologies

The methodologies provides a standardized approach to conduct penetration testing. It's important to remember that attacks don't follow rules or approaches. OSSTMM is an open source tool that tries to describe consistent and repeatable assessments. It proposes different tests depending on the cost and time needed.

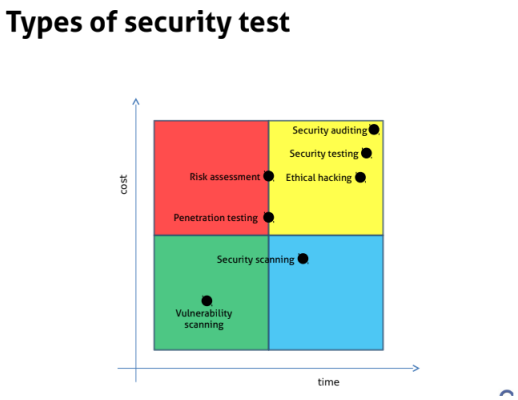


Figure 36: Tests type w.r.t cost and time

Another important differentiation between tests is the prior knowledge of the network by the company, and the knowledge the client has about the details of the test.

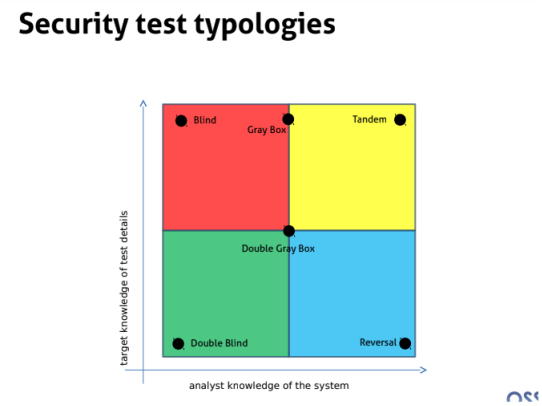


Figure 37: Tests type w.r.t to client and analyst's knowledge

14.8.1 OSSTM types of tests

1. Vulnerability scanning: it's automated, and checks for known vulnerabilities against a system or different systems on the network
2. Security scanning: it's a vulnerability scan which include manual false positive verification, network weakness identification, and customized, professional analysis
3. Penetration testing: a goal-oriented project which simulates an attack from a malicious hacker; it includes gaining privileged access by pre-conditional means.
4. Risk assessment: it's a security analysis through interview and mid-level research which includes business justification, legal justifications and industry specific justifications
5. Security auditing: it's a hands-on, privileged security inspection of the OS and application of a system/s within a network/s
6. Ethical hacking: it's a penetration test whose goal is to discover security flaws in the system and the network within the predetermined project time limit
7. Security testing / posture assessment: it's a project-oriented risk assessment of systems and networks through the application of professional analysis on a security scan where penetration is often used to confirm false positive and false negatives as project time allows.

14.8.2 Type of testing attacks

1. Blind: the analyst engages the target with no prior knowledge of its defenses, assets or channels; the client is prepared for the audit, knowing in advance all of its details. The goodness of this test depends on the skills of the analyst.
2. Double blind (or black box): the same as blind, but the client is not informed about the details of the audit. The outcome depends both on the skill of the analyst and how much the client is prepared against unknown variables.
3. Gray box (vulnerability assessment): the analyst has a limited knowledge about the client's system, meanwhile the client knows details about the auditing. It's often initiated by the target as a self assessment.
4. Double gray box (or white box): the analyst has limited knowledge of the system, and the client is notified about the scope and the time of the audit, but not the channel tested or the test vectors.
5. Tandem (or In-House audit or Crystal box): The analyst and the client both are prepared for the auditing, knowing all the details. This type will test the protection and controls of the client, but not its preparedness to unknown variables.
6. Reversal (or red team exercise): the analyst engages the client knowing fully its system, but the client won't know what, when and how the analyst be testing.

14.9 Vulnerability scanners

Vulnerability scanners discover and enumerate vulnerabilities on systems or application; they don't protect the network from them though.

They are very good at checking for hundreds of potential problems quickly, they are automated and regurarly used.
They can help identify rogue machines, provide a generic risk level, explain why an item is a risk, and provide detailed information on how to apply remediation.
There are different types of scanners: agent-based (agents are monitoring software running on each end node and communicate with a central scanner), or agent-less. They can be passive (limits themselves at monitoring traffic and are unobtrusive), or active (they can be used to simulate attacks.

Network scanners are typically slower than simpler port scanners, and they may disrupt operations (e.g. if a DDoS testing is performed); they make use of vulnerability DB, who must be updated frequently. The security organization often doesn't own the network or the system, and security resources are often decentralized.

Network scanner perform the following steps:

1. Discovery: check to see if target is responding to network traffic
2. Port scans: find what ports are listening on the system
3. Service detection: based on port listening, talk to them to guess what they are based on patterns and other information
4. Operating system detection: based on available information, guess the operative system
5. Vulnerability assessment: based on the available information, determine the vulnerabilities that may exist on the system; often listed by the CVE identifier

The network scanner principles are: they probe specific services/protocols for weakness (not just generic IP addresses like nmap), are most useful when working on pre-gathered info (from databases or other network scanning), and they are similar to virus scanning software (they can't find vulnerabilities that are not in the database).

14.9.1 CVE

Common Vulnerabilities and Exposure is a dictionary of common names for publicly known information security vulnerabilities. It is the industry standard for vulnerability and exposure identifiers; it provide reference points for data exchange so that cybersecurity products and services can speak with each other.

When a new cyber-vulnerability is discovered, a Common Vulnerability and Exposure number is publicly assigned to it by an assigning authority (such as MITRE); the vulnerability is then analyzed by the NIST, whose scoring system (CVSS) provides both a severity score and 8 commonly adopted security attributes of the vulnerability.

Some type of vulnerabilities comprehends: Memory exposure/leakage, bad configurations, bad implementations, default credentials, use of old/legacy protocols.

14.9.2 Vulnerability scan data

The data gathered is just a long spreadsheet style list, so it's impossible to just read over the thousand pages reported by the scanner; there's the need to provide some prioritization: what are the biggest risks? what is the acceptable level of risk? what is the threat level in the industry? The data must be turned into actionable information.

You start with something like this..

CVE	CVSS	Host	Protocol	Port	Name	Synopsis	Description	Solution
CVE-2003-0831	9	192.168.150.131	tcp	21	ProFTPD File Transfer Newline Character Overflow	Arbitrary code may be run on the remote server.	The remote host is running a version of ProFTPD which seems to be vulnerable to a buffer overflow when a user downloads a malformed ASCII file. An attacker with upload privileges on this host may abuse this flaw to gain a root shell on this host. *** The author of ProFTPD did not increase the version number *** of his product when fixing this issue, so it might be false	Upgrade to ProFTPD 1.2.9 when available or to 1.2.8p
CVE-1999-0502	10	192.168.150.131	tcp	22	Default Password (toor) for 'root' Account	An account on the remote host uses a known password.	The account 'root' on the remote host has the password 'toor'. An attacker may leverage this issue to gain total control of the affected system.	Change the password for this account or disable it.
CVE-1999-0103		192.168.150.131	udp	7	Echo Service Detection	An echo service is running on the remote host.	The remote host is running the 'echo' service. This service echoes any data which is sent to it. This service is unused these days, so it is strongly advised that you disable it, as it may be used by attackers to set up denial of services attacks against this host.	- Under Unix systems, comment out the 'echo' line in /etc/inetd.conf and restart the inetd process - Under Windows systems, set the following registry key to 0 : HKLM\System\CurrentControlSet\Services\SimpTCP\Parameters\EnableTcpEcho HKLM\System\CurrentControlSet\Services\SimpTCP\Parameters\EnableUdpEcho Then launch cmd.exe and type : net stop simptcp net start simptcp To restart the service.
CVE-1999-0635		192.168.150.131	udp	7	Echo Service Detection	An echo service is running on the remote host.	The remote host is running the 'echo' service. This service echoes any data which is sent to it. This service is unused these days, so it is strongly advised that you disable it, as it may be used by attackers to set up denial of services attacks against this host.	- Under Unix systems, comment out the 'echo' line in /etc/inetd.conf and restart the inetd process - Under Windows systems, set the following registry key to 0 : HKLM\System\CurrentControlSet\Services\SimpTCP\Parameters\EnableTcpEcho HKLM\System\CurrentControlSet\Services\SimpTCP\Parameters\EnableUdpEcho Then launch cmd.exe and type : net stop simptcp net start simptcp To restart the service.
							When contacted, chargen responds with some random characters (something like all the characters in the alphabet in a row). When contacted via UDP, it will respond with a single UDP packet. When contacted via TCP, it will continue spewing characters until the client closes the connection.	- Under Unix systems, comment out the 'chargen' line in /etc/inetd.conf and restart the inetd process

Figure 38: Vulnerability scan raw data

First of all, data must be analyzed and context must be added; then priorities should be made: what is mort important for the target company? CVSS scores are a baseline for severity, but risks and how sever they are vary by company.

Then actions sould be make: remediate or fix (it's impossible to fix everything); it can also be useful to ask why vulnerabilities are here in the first place.

Common system vulnerability scanners are: Nessus, Retina, Nmap, OpenVas, etc..

Application scanner tools are: IBM appscan, Nessus, sqlmap, watobo, etc..

14.9.3 OpenVAS

It's the open source fork of nessus v2, so it's similar to nessus: can enumerate multiple OSs, identify a computer on a network by using port scanning and zone transfers, identify OSs by conducting port scanning, identify via enumeration any logon accounts, learn names of shared folders by using enumeration, and identify services running.

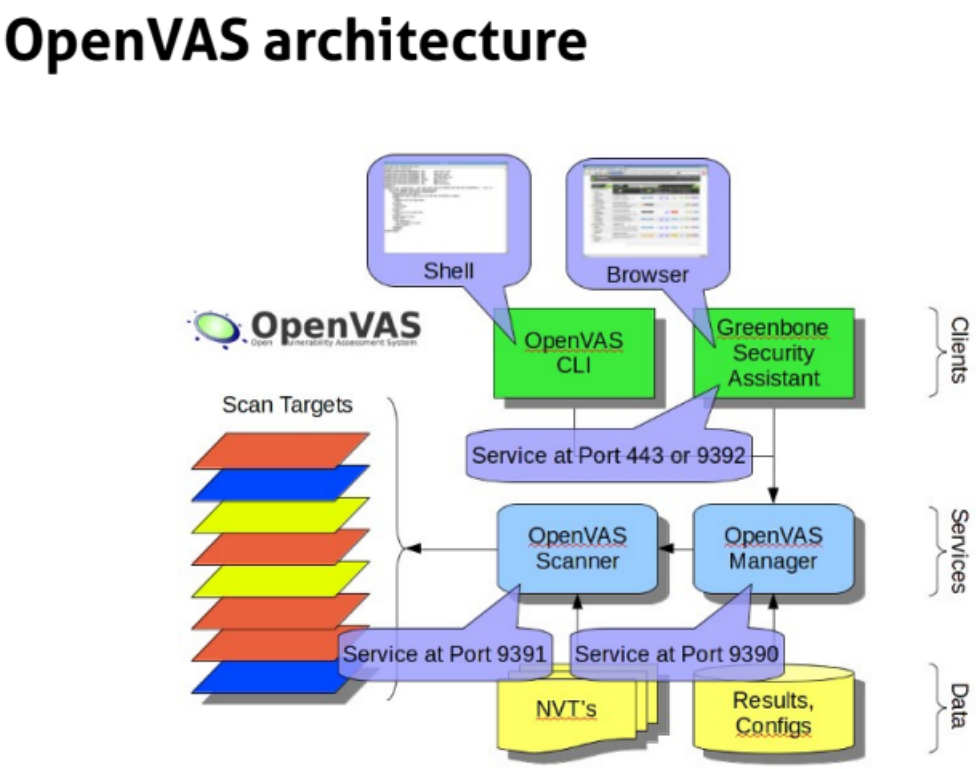


Figure 39: Openvas architecture

Its components are: the scanner, who scans and receive signature updates of NVTs, network vulnerability tests, the manager, who manages scanning rules, consolidates reports, and stores setting and history based sqlite, and the clients, command line and web GUI (greenbone secuirty assistant).