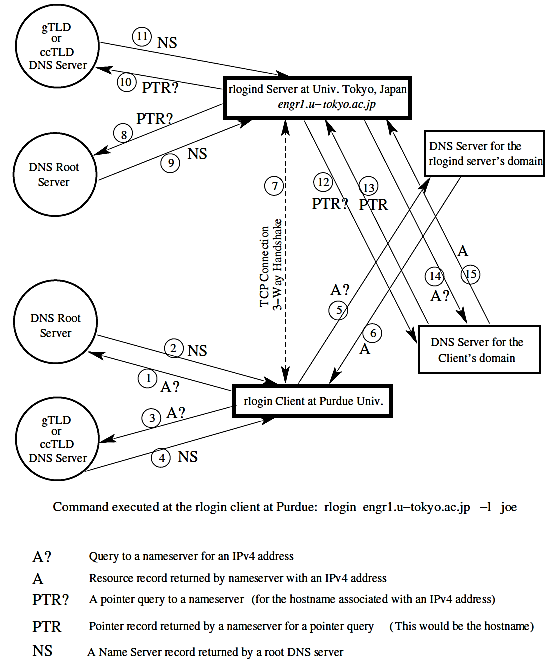
▓DNS(Domain Name Service / Server / System / Space): translate symbolic hostnames into the numerical IP addresses and vice versa. Lists mail exchange servers that accept email for different domains.

* hostnames and IP addresses do not necessarily match on a one-to-one basis.
  + Many hostnames may correspond to a single IP address: allows virtual hosting, a single machine to serve many web sites
  + a single hostname may correspond to many IP addresses: facilitate fault tolerance and load distribution.
* MTAs (Mail Transfer Agents): ex. sendmail use DNS to find out where to deliver email for a particular address. The domain to mail-exchanger mapping is provided by MX records stored in DNS servers.
* DNS is distributed databases: serving billions of DNS requests daily for IP addresses and mail-exchange hosts.
* DNS is an open and openly extendible database: anyone can set up a DNS server and plug it into the network of worldwide network of DNS servers.
* DNS Hijacking on Non-Existent Domain Names: instead of sending NXDOMAIN (non-existent domain) error message to your browser, the ISPs DNS server sends back a browser redirect to an advertisement-loaded website or suggestions for domains that are similar to what your browser is looking for.
* name resolver program: get the IP address associated with a symbolic hostname, or the other way around
* private home network:
  + four or five machines: create a host table (in the /etc/hosts) file on each machine, name resolver program would then consult this table to determine the IP address of each machine in the network.
  + More machines: install a DNS server in the network.
* /etc/host.conf tells the system in what order it should search through the following two sources of hostnames-to-ipaddress mappings: /etc/hosts and DNS
* DNS hijacking: malware that you may have inadvertently downloaded by clicking on a URL in a spam email may overwrite the entries in the file /etc/resolv.conf. This would cause your name resolution requests to be serviced by a rogue DNS. Your browser may end up visiting a malicious website that is made to look like the one you were actually trying to reach, you could end up giving your personal information, such as your bank account information
* gethostbyname() and getaddrinfo() translate the symbolic hostnames into IP addresses. gethostbyaddr() and getnameinfo() carry out reverse name lookup
* DNS system is organized in a hierarchical fashion.
  + top of the hierarchy are the 13 root servers. The IP addresses of these root servers are programmed into every name resolver so that it never has to query anyone for the IP addresses of the root servers.
    1. Generic Top Level Domain DNS server (gTLD): If a root server receives a query for .com domain, the root server sends back the IP address of one or more gTLD nameservers in charge of the .com domain.
    2. Country Code Top Level Domain DNS server (ccTLD): if a root server receives a query for .jp domain, the response back from root consists of the IP address of the ccTLD server in charge of the .jp domain.
* simple login connection with a remote host: a connection that involves no exchange of security related information (Messages1~6 constitute iterative namelookup for the numerical IP address associated with a domain namenor a hostnname.)



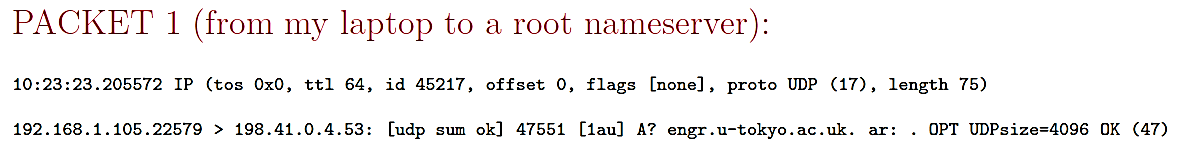
1. on the client side first enters the rlogin command (a Unix program that allows users to log in on another host using a network).
2. client resolver first contacts one of the root nameservers for where to go for resolving the names that end in .jp
3. root nameserver responds back with the IP address of the ccTLD DNS server in charge of the top-level .jp domain.
4. client contacting the ccTLD nameserver for the jp domain.
5. DNS server responds back with the IP address for the authoritative nameserver for the /u-tokyo.ac.jp domain.
6. client contacting the nameserver for the u-tokyo.ac.jp domain.
7. nameserver responds back with the desired IP address.
8. client TCP send a SYN packet to the server for initiating the connection. The server engage in a 3-way handshake to complete a TCP circuit.
9. server wants to know the hostname identity of the client that has connected with it. So the server sends a pointer query (server wants to carry out a reverse DNS lookup) to one of the root servers that may be different from the root server used by the client.
10. The root nameserver responds back with the IP address of the gTLD or the ccTLD nameserver that is relevant to the numerical address in the pointer query.
11. the client contacting the ccTLD nameserver for in-addr.arpa domain relevant to the numerical IP address in question.
12. DNS server responds back with the IP address for the authoritative nameserver for the more specific in-addr.arpa nameserver relevant to the pointer query.
13. rlogind server sends the same pointer request to the domain-specific nameserver whose IP address was received in message 7.
14. server obtains the fully qualified domain name (FQDN) of the client.
15. to account for the possibility that the nameserver for the in-addr.arpa domain (that is used for reverse lookups) may not be the same as the regular nameserver on the client side, the rlogind server sends an A query for the IP address associated with the FQDN it retrieved in message 13.
16. Message 15 then supplies the IP address associated with symbolic hostname for the client.
17. Rlogind server then compares this IP address with IP address in TCP connection that is marked as 7 in Figure 1. If the IP addresses are the same, server allows the client to connect, assuming that the client has the login privileges at the server.

most commonly a DNS

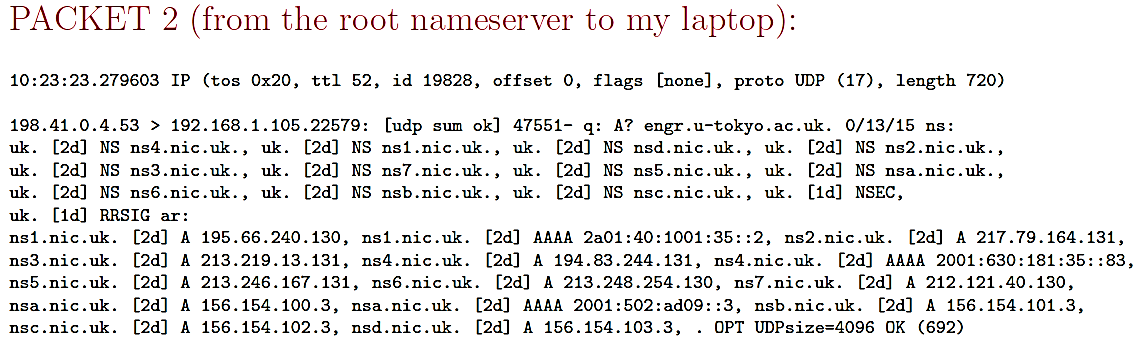
request for name lookup is sent out in the form of a UDP

packet.

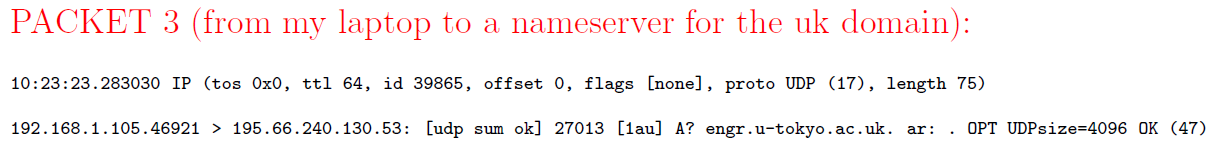
* most commonly a DNS request for name lookup is sent out in a UDP packet.
* UDP packet consists of an 8 byte header:
  + 2 bytes for the source port
  + 2 bytes for the destination port
  + 2 bytes for the length of the packet, includes the length of the header
  + 2 bytes for the checksum.
* Since there is no expectation of a return answer to UDP broadcasts. there would be no point in including the source port info in the response packet. When a UDP packet descends through the IP layer in the TCP/IP protocol stack, it picks up through the IP header the IP address of the destination.
* Port 53 is the port on which a DNS server listens to the incoming name lookup requests and through which it provides its answers.
* laptop figures out that the hostname given to the ssh command does NOT exist
  + 192.168.1.105.22579 > in the first packet says that my laptop, whose 198.41.0.4.53 IP address is 192.168.1.105, is using the ephemeral port 22579 to send a UDP packet to the root server whose IP address is at its port 53, which is the standard port 198.41.0.4 assigned to DNS servers.
  + Transaction ID of a DNS query 47551 is a 16-bit randomly generated integer, making it more difficult to mount a DNS cache poisoning attack. A valid answer to a DNS query must contain the same integer.
  + string A? engr.u-tokyo.ac.uk means that laptop is requesting the IPv4 address for the hostname engr.u-tokyo.ac.uk.



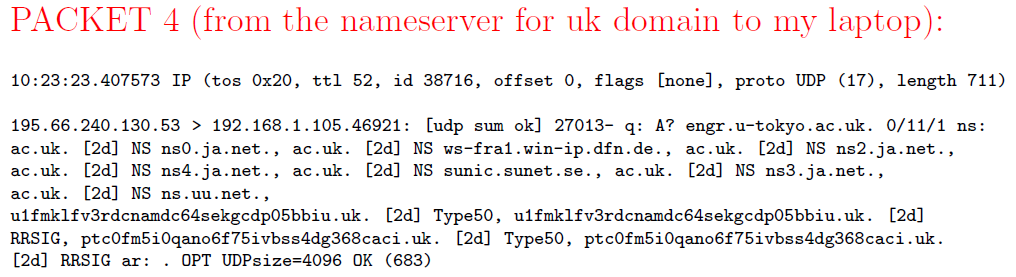
* + The answer returned by the root nameserver consists of the symbolic names and subsequently the IPv4 addresses for several nameservers responsible for the uk domain. For example, one of the nameservers listed for the uk domain is ns1.nic.uk and its IPv4 address is 195.66.240.130
  + A string such as ns1.nic.uk.[2d] A195.66.240.130 is a Resource Record [2d] is TTL, associated with mapping between the symbolic hostname and the IP address ns1.nic.uk 195.66.240.130 is two days.

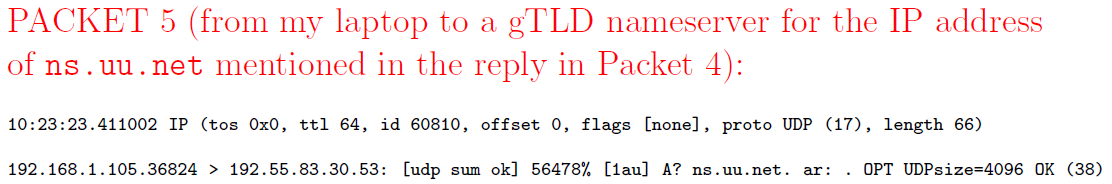


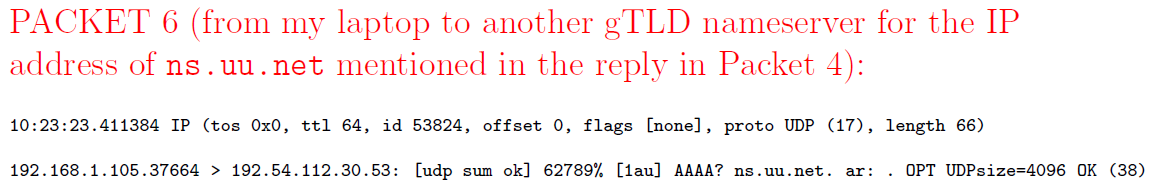
* + String 192.168.1.105.46921 > 195.66.240.130.53 tells us that this is a packet from my laptop to the nameserver for the ns1.nic.uk uk top-level domain. Transaction ID number is 27013.



* + nameserver responds back by sending to my laptop the symbolic hostnames for several nameservers for the ac.uk subdomain.

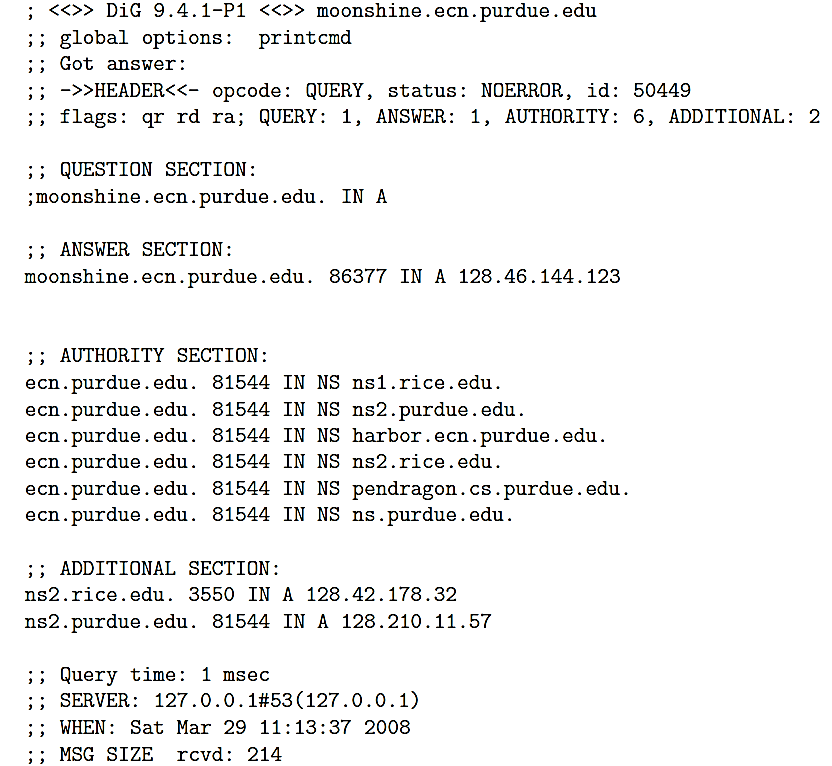




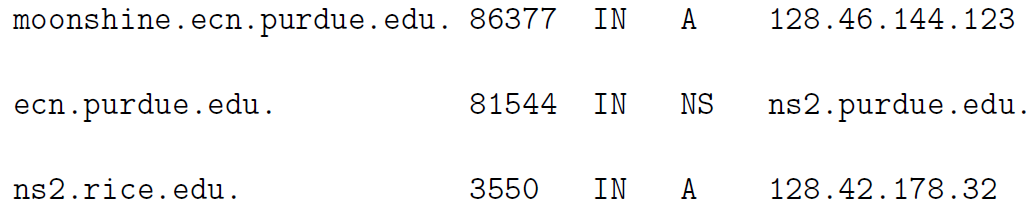


* each host is represented in DNS by two DNS records: an address record and a reverse mapping pointer record.
* fully qualified domain names (FQDN): The root domain is represented by a period ”.” and formal DNS name of the purdue.edu domain is purdue.edu. period at the end stands for the root of the DNS tree.
* For the Domain Name System, all of the internet is divided into a tree of zones. Each zone, consisting of a Domain Name Space served by a DNS nameserver that consists of two parts:
  + an Authoritative Nameserver for the IP addresses for which the zone nameserver directly knows the hostname-to-IP address mappings;
  + a Recursive Nameserver for all other IP addresses.
* zone file: authoritative nameserver file that contains the mappings between the hostnames and the IP addresses

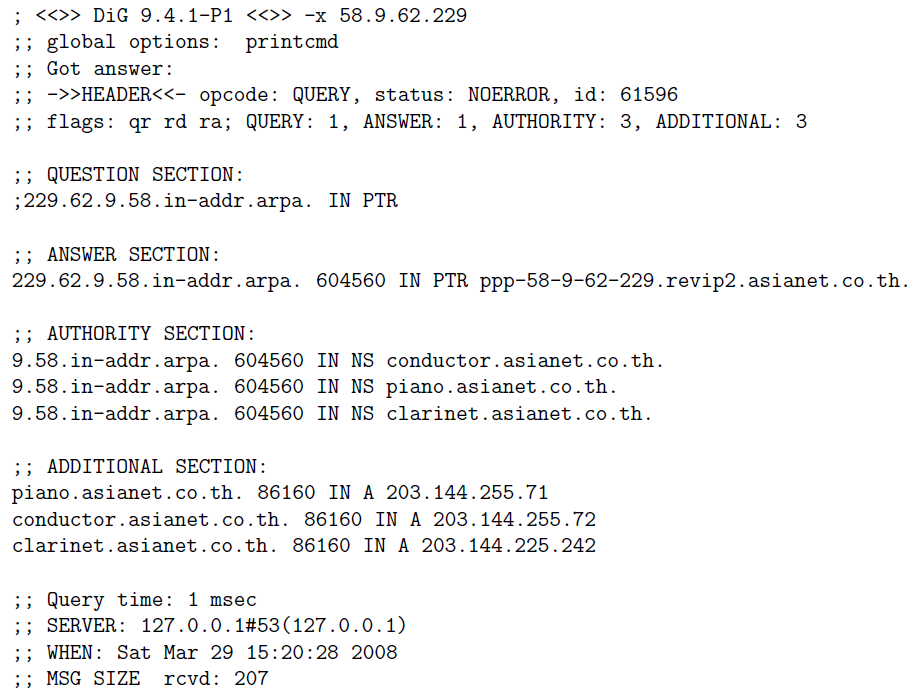
▓dig (domain information groper): is a useful command for interrogating DNS nameservers for information about the host IP addresses, mail exchanges, nameservers for other domains, and so on. Numbers like 86377, 81544, 3550 are TTL in seconds. (EX. dig moonshine.ecn.purdue.edu)



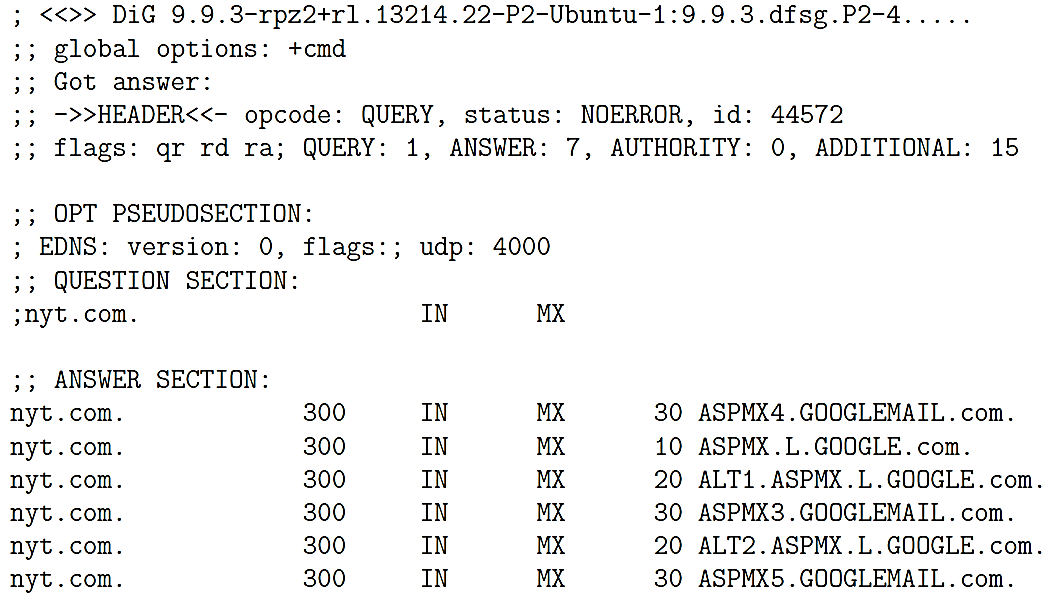
* AUTHORITY SECTION: lists nameservers for the ecn.purdue.edu domain.
* Additional Section: lists the IP addresses of the nameservers named in the Authority Section.
* Resource Record (RR) consists of 5 items:

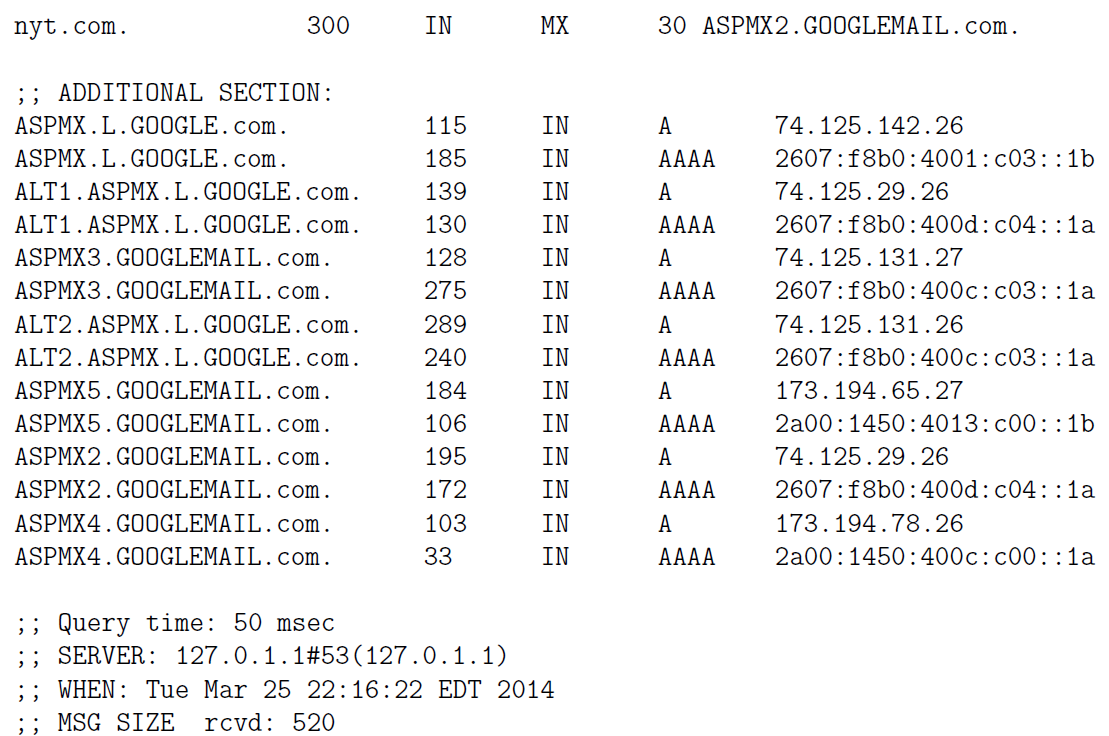


* fully qualified domain name (FQDN): such as ns2.rice.edu.
* Time-to-live (TTL): such as 86377 seconds
* Class of the record: such as IN stands for class internet
* Type of the record.
  + A: address record in the form of an IPv4 numerical address.
  + AAAA: address record in the form of an IPv6 numerical address. AAAA indicate that an IPv6 address is four times the size of an IPv4 address.
  + NS: nameserver record consisting of name(s) of nameserver(s) that can be queried for resolving a given hostname.
  + PTR: pointer record that is symbolic hostname associated with a numerical IP address. It is returned in reverse name lookup.
  + MX: a mail exchange server for a given host.
* record data: such as the IPv4 address 128.46.144.123
* dig -x 58.9.62.229 do reverse DNS lookup. Query string has the four integers of the IP address in the reverse order and the string ends in the suffix in-addr.arpa.

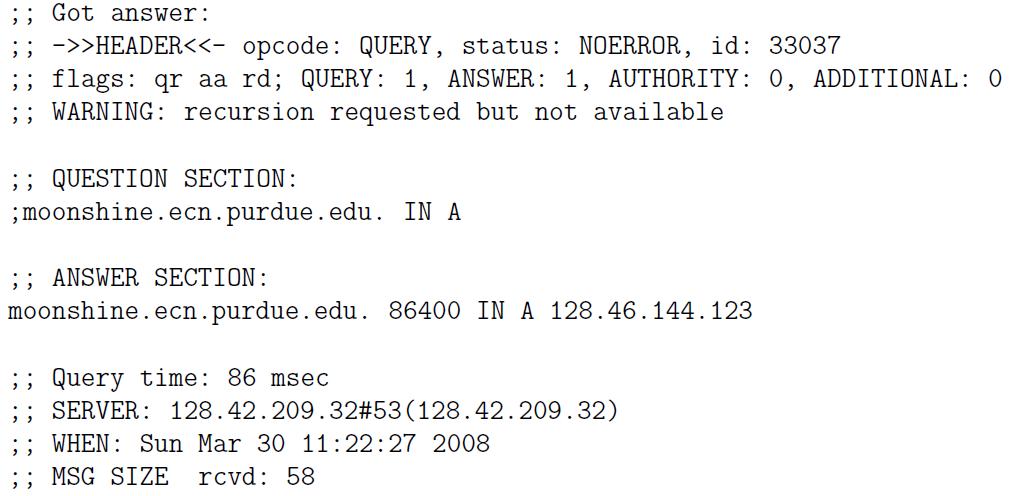


* dig nyt.com MX: the IP address of the host (or hosts) responsible for mail exchange for a domain nyt.com
  + when multiple hosts are returned for the mail exchange service for a domain and each has its own MX preference number, the MX hosts with the smallest preference numbers must be tried first for mail exchange before those with higher numbers are attempted.



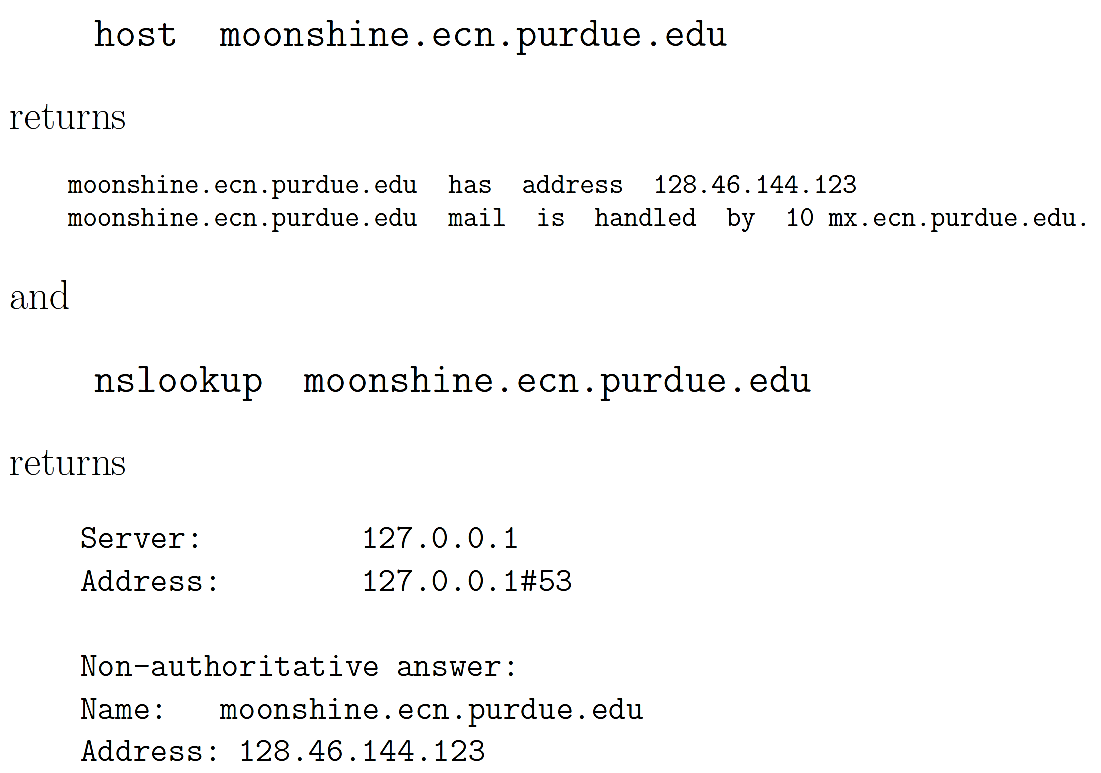


◆dig @ns1.rice.edu +nocmd moonshine.ecn.purdue.edu: ask the DNS server running at Rice University for the IP address for moonshine.ecn.purdue.edu:



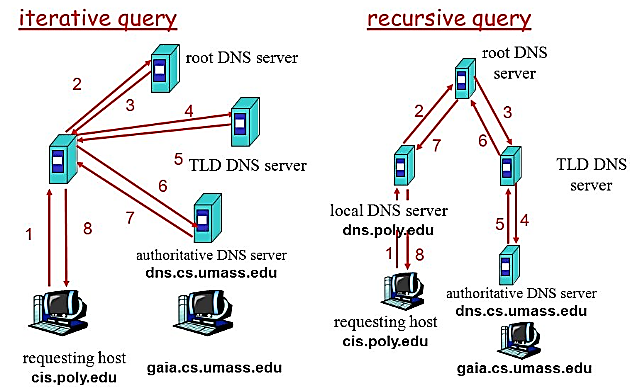
* dig examines the contents of your /etc/resolv.conf file for the nameservers to send the query to the nameservers in the order they are listed in the /etc/resolv.conf file.

▓host and nslookup are the other utilities that can also be used to query nameservers.



▓When a zone administrator A (purdue.edu) wants to let another administrator B (robotics.purdue.edu,) control a part of that zone, A can delegate control for that subdomain to B.

1. create a nameserver with a name like ns.robotics.purdue.edu. This nameserver would become the SOA (Start of Authority) for all the hostnames within the domain . robotics.purdue.edu
2. main nameservers for purdue.edu would be authoritative nameservers for all hostnames within the purdue.edu domain but not including the hostnames in robotics.purdue.edu. With respect to the hostnames in robotics.purdue.edu, the main purdue.edu nameservers would be the recursive nameservers.



▓master and slave nameservers (primary and secondary nameservers.):

* A primary nameserver is the default for a name lookup. A query will failover to the secondary (or to the tertiaries) if the primary is not available.
* Zone Transfer: Any changes to the nameserver record for a local domain would be made to the master nameserver and would then get automatically synced over to a slave via what is referred to as a

▓DNS CACHE

* it is not your computer as a single entity that carries out a DNS name lookup. It is a client application such as the Internet Explorer, Firefox, a mail client such as sendmail, etc., that sends a query to a DNS nameserver.
* you are within the purdue.edu domain and you point your browser to www.nyt.com, the browser will send that URL to one of the nameservers of the purdue.edu domain.
  + If this is the first request for this URL received by the nameserver for purdue.edu, the nameserver will forward the request to the nameserver for the com domain
  + if this was not the first request for the name resolution of www.nyt.com, it is likely that the local nameserver would be able to resolve the URL by looking into its own cache.
* the various client applications maintain their own DNS caches usually with very short caching times
* operating system would be programmed to look up the information in /etc/hosts for any direct hostname-to-IP address mappings you might have placed there. operating system may also maintain a local cache for the previously resolved hostnames with relatively short caching times

▓TTL Time Interval

* The TTL value associated with a hostname is set by the administrator of the authoritative DNS server that returns the IP address along with its TTL.
* While DNS caching makes the hostname resolution faster, any changes to the DNS do not always take effect immediately and globally.
* authoritative nameserver is publishing nameserver recursive nameserver is caching nameserver
* making the DNS system secure against a large-scale Denial-of-Service attacks
  + even if the root servers were to be taken down by an adversary, the information about the TLD would continue to reside in the lower-level nodes of the DNS tree of zones for roughly two days. That would be long enough for remedial action to be taken against the adversary.
  + if an adversary took down the gTLDs and the ccTLDs the slave servers for those would provide immediate

▓BIND (Berkeley Internet Name Daemon) is the most commonly used implementation of a domain name server (DNS).

▓DNS cache poisoning attack: poison the cache of the nameserver running on the machine harbor.ecn.purdue.edu by placing in its cache an incorrect IP address for, say, the

amazon.com domain.

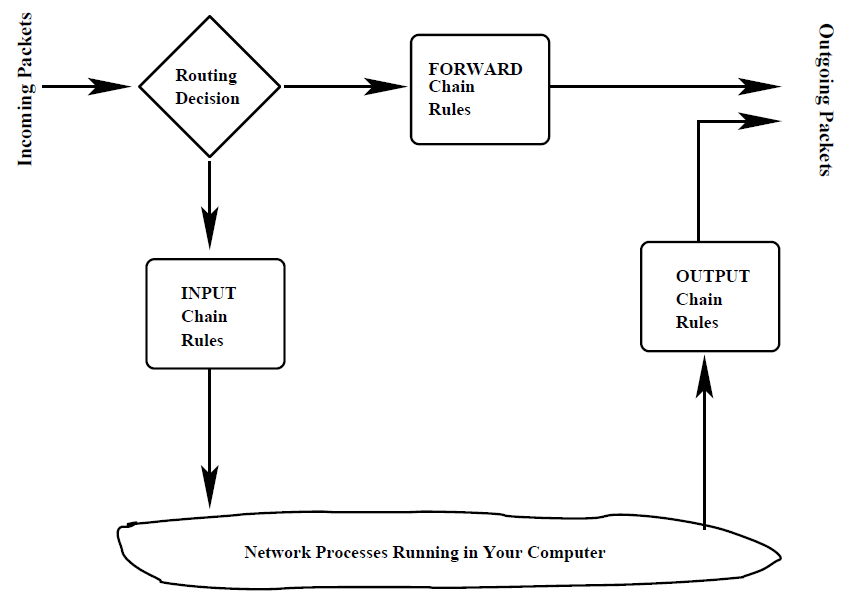
1. asking the DNS server running at harbor.ecn.purdue.edu to carry out the name lookup for the domain amazon.com by dig amazon.com @harbor.ecn.purdue.edu
2. DNS server will make an NS query to nameserver in charge of the com top-level domain for the IP addresses of the nameservers in charge of the amazon.com domain. This NS query will contain a pseudorandom Transaction ID integer.
3. As execute the dig command in one window of your machine, in another window you will simultaneously fire up a script that floods harbor.ecn.purdue.edu with manually crafted packets that look like the reply the DNS server at harbor is expecting but that contain the wrong IP address.
4. Each reply will contain a different Transaction ID integer, with the hope that the Transaction ID in one of those fake replies will match the Transaction ID in the query sent out by harbor. The attacker can safely assume that the port in the destination address used in the query packets issued by the attacked nameserver is 53 since that is the standard port monitored by nameservers.
5. there is now a race between the correct reply from the nameserver that has the legitimate IP address for the amazon.com domain and the flood of fake replies sent by you the attacker.

▓another weakness of the DNS protocol itself is a caching nameserver accepting resource records for hosts not asked for in the query. DNS nameservers that are purely authoritative are not vulnerable to his attack. However, for a DNS server to be useful, it can be authoritative only with respect to the names that are in the domain of the server. With respect to all other names, a nameserver that is otherwise authoritative must serve as a

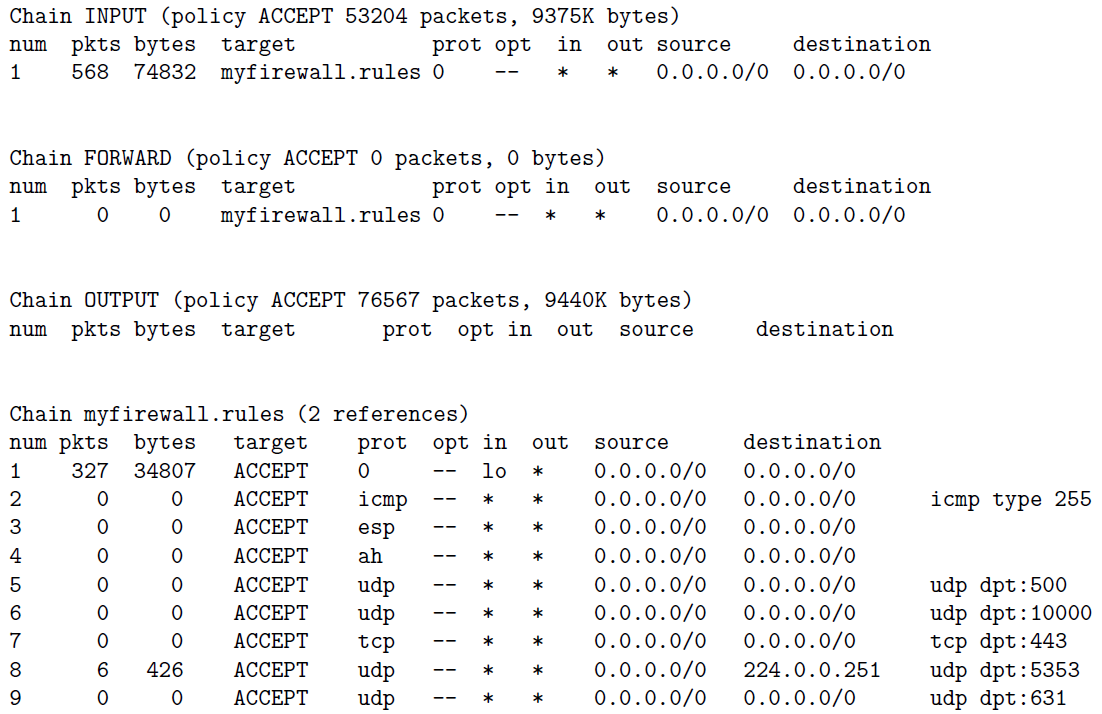
* recursive nameserver that allows caching for the sake of efficiency in name lookup. Attacker want to place in the cache of the nameserver ns.purdue.edu a fake IP address for [www.foo.com](http://www.foo.com).
* attacker starts by querying nameserver for purdue.edu domain for possibly nonexistent symbolic hostnames 1.foo.com, 2.foo.com etc. The nameserver ns.purdue.edu will have no entries for this hostnames and first contact one of the root nameservers for the com domain and will eventually contact the nameserver for the foo.com domain for the IP addresses for 1.foo.com, 2.foo.com, etc.
* attacker now sends spoofed replies from ns.foo.com to ns.purdue.edu for all of the queries emanating from the latter for the various versions of foo.com hostnames. Attacker will have to race against the true answers being sent to ns.purdue.edu from the authentic ns.foo.com.
* a caching nameserver such as ns.purdue.edu would not only accept the Resource Records in the Answer Section of the fake replies to its queries, but also the RRs in the Additional Section where the attacker may even place a fake address for ns.foo.com. The attacker could also associate a long TTL with this entry.
* Subsequently, any third-party accessing the ns.purdue.edu nameserver for an IP address for any host in the foo.com domain will reach the attacker nameserver instead of the true nameserver for the foo.com domain. Now the attacker could create any set of hostname-to-IP address mappings for the hosts in the foo.com domain.
* fix for the problem
  + Make it more difficult to take advantage of birthday paradox for guessing the Transaction ID in a query emanating from a resolver or a recursive nameserver.
  + insisting that all recursive nameservers carry out bailiwick check of the RRs in the replies sent by the other nameservers before accepting them. Bailiwick check means to not accept an RR if it contains a hostname that was not in the outgoing query. In this manner, even if the attacker managed to corrupt the cached IP addresses for specific hostnames such as 1.foo.com, 2.foo.com, etc., the attacker will not be able to corrupt the entry for the nameserver ns.foo.com at the same time.

▓FIREWALLS

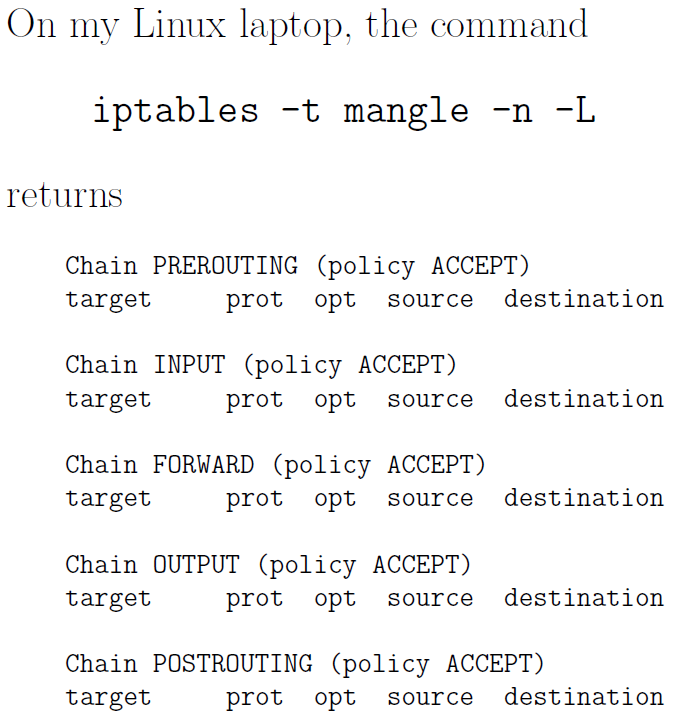
* Two primary types of firewalls
  + packet filtering firewalls
  + proxy-server firewalls: all network traffic meant for the computers in a LAN is routed through a proxy server that allows the proxy server to exercise access control over the traffic
* Linux kernel uses four tables, each consisting of rule chains, for processing the incoming and outgoing packets. Each packet is subject to each of the rules in a chain and the fate of the packet is decided by the first matching rule.:
  + filter table: contains at least three rule chains (built-in chains that cannot be deleted)
    - INPUT: for processing all incoming packets. When a packet comes in the kernel first looks at the destination of the packet. (labeled as routing). If routing decision is that the packet is intended for the machine in which packet is being processed, packet passes to the INPUT chain. If incoming packet is destined for another machine, packet goes to FORWARD chain.
    - OUTPUT: If a program running on computer wants to send a packet out of the machine, the packet must traverse through the OUTPUT chain of rules.
    - FORWARD: for processing all packets being routed through the machine. packet forwarding only occurs when the machine is configured as a router.



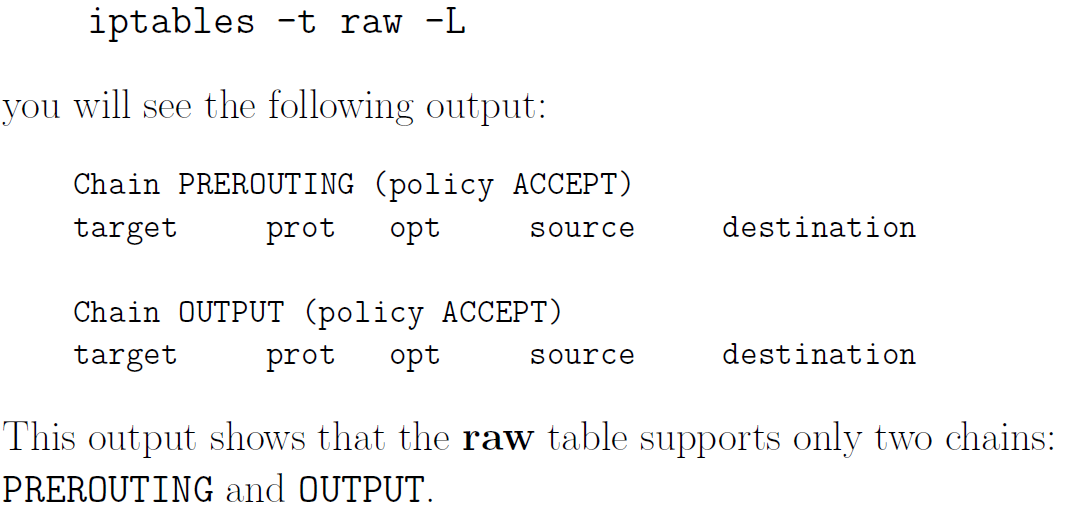
* If a packet reaches the end of a chain, Linux kernel looks at the chain policy to determine the fate of the packet. In a security-conscious system, this policy usually tells the kernel to DROP the packet.
  + nat table (Network Address Translation): When your machine acts as a router, it would need to alter either the source IP address in the packet passing through, or the destination IP address, or both. Nat table consists of four built-in chains:
    - PREROUTING: for altering packets as soon as they come in,
    - INPUT: for altering the incoming packets after they have been subject to pre-routing rules if any,
    - OUTPUT: for altering locally-generated packets before routing
    - POSTROUTING: for altering packets as they are about to go out.
  + mangle table: used for specialized packet alteration, has five rule chains:
    - PREROUTING: for altering incoming packets before a routing decision is made concerning the packet,
    - OUTPUT: for altering locally generated outgoing packets,
    - INPUT: for altering packets coming into the machine itself,
    - FORWARD: for altering packets being routed through the machine
    - POSTROUTING: for altering packets immediately after routing decision.
  + raw table: used for configuring exceptions to connection tracking rules. when a raw table is present, it takes priority over all other tables.
    - connection tracking rules: packets coming into your computer from a host in the internet through a connection that you initiated.
* a packet coming in for the local host will first be seen by the mangle.PREROUTING, nat.PREROUTING, and mangle.INPUT chains before it is seen by the filter.INPUT chain. The
* mangle.PREROUNTING and the nat.PREROUTING chains would decide whether the packet should be sent to a local process or forwarded to another communication interface. If the former is the case, the mangle.INPUT could subsequently mark the packet for downstream security based examination before the packet is subject to the rules in the filter.INPUT chain.
* STRUCTURE OF THE filter TABLE
  + num : The rule number in a chain.
  + pkts : The packet count processed by a rule so far.
  + bytes : The byte count processed by a rule so far.
  + target : The action part of a rule. The target can be one of the following:
    - ACCEPT
    - DROP: drop the packet without sending an error message to the originator of that packet.
    - REJECT: drop the packet and the sender is sent an error message that depends on the argument supplied to the originator of that packet.
    - REDIRECT: send the packet to a new destination (used with NAT).
    - RETURN: return from this chain to the calling chain and to continue examining rules in the calling chain where you left off. When RETURN is encountered in a built-in chain, the policy associated with the chain is executed.
    - the name of the chain to jump to
  + proto : The protocol associated with the packet to be trapped by this rule. The protocol may be either named symbolically or specified by a number. Each standard protocol has a number associated with it. The protocol numbers are assigned by Internet Assigned Numbers Authority (IANA).
    - when proto column mentions a user-defined service as opposed to a protocol, then the last column must mention the port specifically.
    - for packets corresponding to standard services, the system can figure out the ports from the entries in the file /etc/services.
  + opt : optional
  + in : The input interface to which the rule applies. Figure example: Since the input interface mentioned is lo and since no ports are mentioned, this rule applies only to the packets generated by the applications running on the local system. Therefore, with this rule, you can request any service from your local system without the packets being denied.



* + out : The output interface to which the rule applies. Figure example: There are no rules in this chain. Therefore, for all outbound packets, the policy associated with the OUTPUT chain will be used. This policy says ACCEPT, implying that all outbound packets will be sent directly, without further examination, to their intended destinations.
  + source : The source address(es) to which the rule applies.
  + destination : The destination address(es) to which the rule applies.
  + the last column, with no heading, contains ancillary information related to a rule. It may mention a port, the state of a packet, etc.
  + Policy declaration associated with each chain. The policy sets the fate of a packet and it is not trapped by any of the rules in a chain.
* STRUCTURE OF THE nat (Network Address Translation) TABLE: nat table is used only for translating either packets source address field or its destination address field. nat table allows a host or several hosts to share the same IP address. NAT server receives the packet, rewrites the source and/or destination address and then recalculates the checksum of the packet. Only the first packet in a stream of packets hits this table. After that, the rest of the packets in the stream will have this network address translation carried out on them automatically.
  1. targets for nat table (meaning the actions that are permitted for the rules) are:
     + DNAT: mainly used in cases where you have a single public IP for a local network in which different machines are being used for different servers. When a remote client wants to make a connection with a local server using the publicly available IP address, youd want your firewall to rewrite the destination IP address on those packets to the local address of the machine where the server actually resides.
     + SNAT: mainly used for changing the source address of packets. When a server residing on one of the local machines responds back to the client, initially the packets emanating from the server will bear the source address of the local machine that houses the server. But as these packets pass through the firewall, youd want to change the source IP address in these packets to the single public IP address for the local network.
     + MASQUERADE: Whereas SNAT will substitute a single previously specified IP address for the source address in the outgoing packets, MASQUERADE can substitute a DHCP IP address (that may vary from connection to connection) and takes a little bit more overhead to compute.
     + REDIRECT
* STRUCTURE OF THE mangle TABLE: used for specialized packet alteration, such as for changing the TOS (Type of Service) field, the TTL (Time to Live) field, etc., in a packet header. Following targets can only be used in the mangle table.
  1. TOS: Used to change the TOS (Type of Service) field in a packet.
  2. TTL: used to change the TTL (Time To Live) field of the packet.
  3. MARK: used to give a special mark value to the packet. Such marks are recognized by the iproute2 program for routing decisions.
  4. SECMARK: This target sets up a security-related mark in the packet. Such marks can be used by SELinux fine-grained security processing of the packets.
  5. CONNSECMARK: This target places a connection-level mark on a packet for security processing.

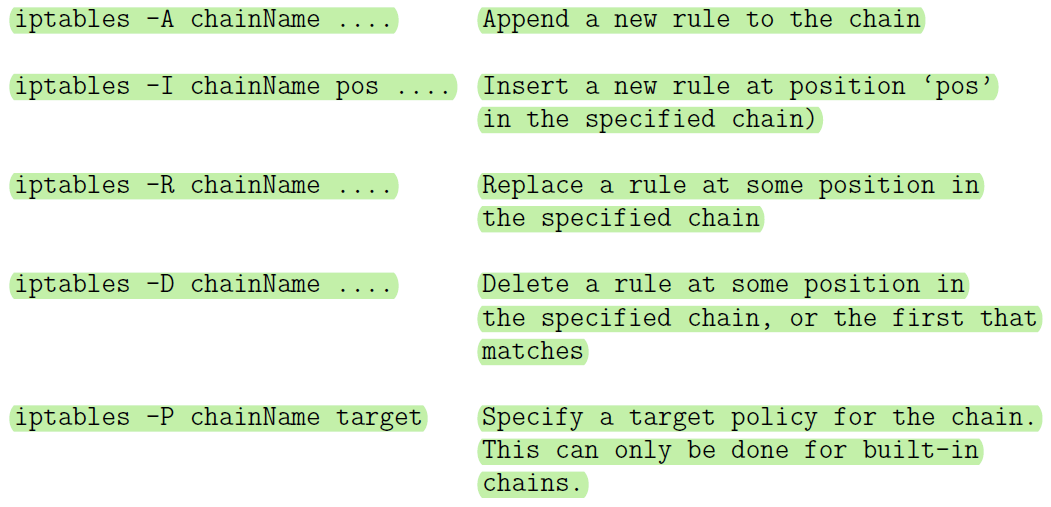
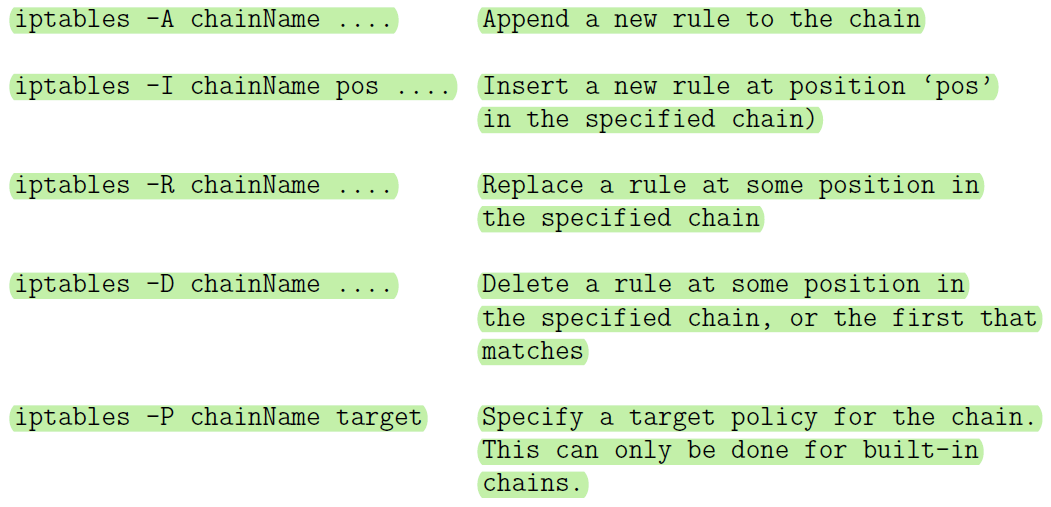


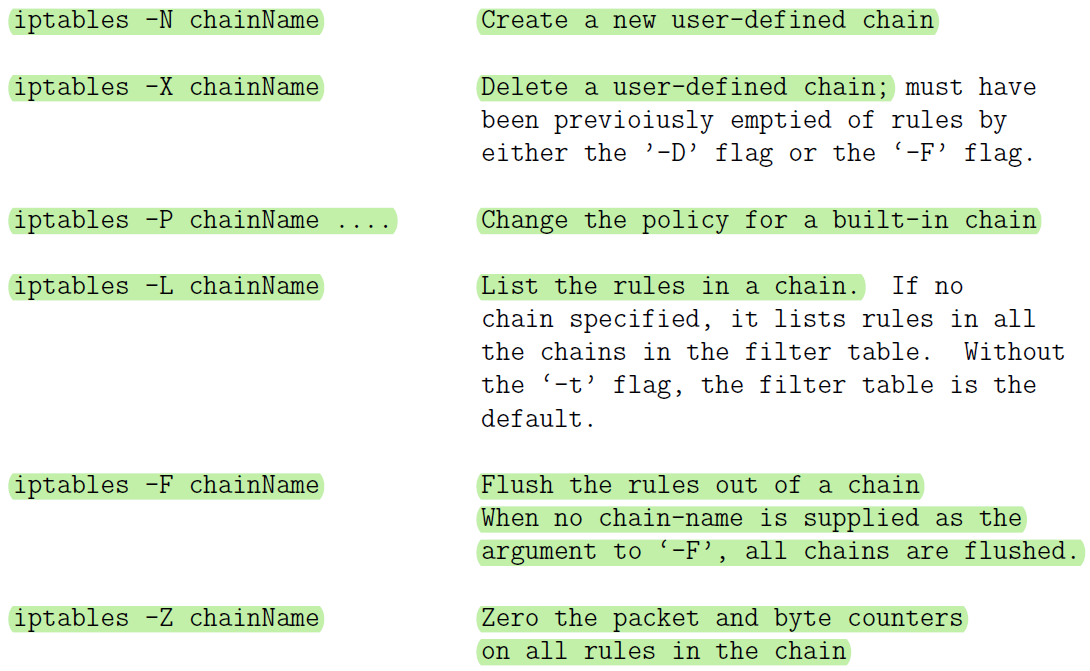
* STRUCTURE OF THE raw TABLE: used for specifying the exemptions from connection tracking that we will talk about later. When rules are specified for the raw table, the table takes priority over the other tables.



▓iptables are created by the iptables command that is run as root with different options.

* optional flags for the iptables command:





* additional flags describe the filtering specifications for each rule: After the first level flags shown above that name a chain, if this flag calls for a new rule to be specified (such as for -A flag) you can have additional flags that specify the state of the packet
  + -p args: for specifying the protocol (tcp, udp, icmp, etc)
  + -s args: for specifying source address(es)
  + --sport args: for specifying source port(s)
  + -d args: for specifying destination address(es)
  + --dport args: for specifying destination port(s)
  + --icmp-type typename: for spcifying the type of ICMP packet
  + -f: For specifying that a packet is a second or a further fragment.
  + -j args: the name of the target to execute when the rule matches
  + -I args: for naming the input interface
  + -o args: for specifying an output interface
  + --syn :To indicate that this rule is meant for a SYN packet.
  + -m match: (This is referred to as a rule seeking an extended match
  + -n: (This forces the output produced by the -L flag to show numeric values for the IP addresses and ports.)
* specification by inversion: Many rule specification flags (such as -p, -s, -d, -f syn, etc.) can have their arguments preceded by ! (that is pronounced not) to match values not equal to the ones given.
  + For the -f option flags, the inversion is done by placing ! before the flag, as in

! -f The rule containing the above can only be matched with the first fragment of a fragmented packet.

* The source (-s, source or src) and destination (-d,destination or dst) IP addresses can be specified in four ways:
  + use the full name, such as localhost or [www.linuxhq.com](http://www.linuxhq.com).
  + specify the IP address such as 127.0.0.1.
  + specification of a group of IP addresses with the notation 199.95.207.0/24 where the number after the forward slash indicates the number of leftmost bits in the 32 bit address that must remain fixed. Therefore, 199.95.207.0/24 means all IP addresses between 199.95.207.0 and 199.95.207.255.
  + uses the net mask directly to specify a group of IP addresses. What was accomplished by 199.95.207.0/24 above is now accomplished by 199.95.207.0/255.255.255.0.
* If nothing comes after the forward slash in the prefix notation for an IP address range, the default of /32 (which is the same as writing down the net mask as /255.255.255.255) is assumed. Both of these imply that all 32 bits must match, implying that only one IP address can be matched.
* /0 means every IP address.
* iptables -A INPUT -s 0/0 -j DROP will cause all incoming packets to be dropped. But note that -s 0/0 is redundant here because not specifying the -s flag is the same as specifying -s 0/0 since the former means all possible IP addresses.

▓A modern iptables-based firewall understands the notion of a stream. This is made possible by the connection-tracking feature of iptables. Connection tracking is based on the notion of the state of a packet.

* States are known through the connection tracking system, which keeps track of all the sessions.
* If a packet is the first that the firewall sees or knows about, it is considered to be in state NEW (SYN packet in a TCP connection)
* if it is part of an already established connection or stream that the firewall knows about, it is considered to be in state ESTABLISHED.
* Connection tracking is also used by the nat table and by its MASQUERADE target in the tables.
* extension modules is one of those extensions to iptables that makes it possible to carry out connection tracking. An extension module is specified by the -m option, a most useful extension module is state. This extension tries to interpret the connection-tracking analysis produced by the ip\_conntrack module.



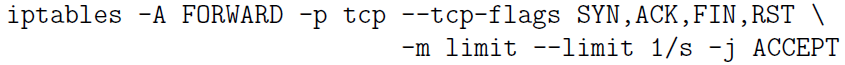
* how the interpretation of the results on a packet produced by the ip\_conntrack module should be carried out is specified by the additional ‘—state’ option supplied to the ‘state’extension module.
* --state suboption supplies a comma-separated list of states of the packet that must be found to be true for the rule to apply, and as before, the ! flag indicates not to match those states. These states that can be supplied as arguments to the --state option are:
  + NEW: A packet which creates a new connection.
  + ESTABLISHED: packet belongs to an existing connection (reply packet)
  + RELATED: packet related to, but not part of, an existing connection, such as an ICMP error, or a packet establishing an ftp data connection.
  + INVALID: A packet which could not be identified includes running out of memory and ICMP errors. Generally these packets should be dropped.
* applies to all packets being forwarded through the ppp0 interface. If such a packet is NOT requesting a new connection, it should be dropped.



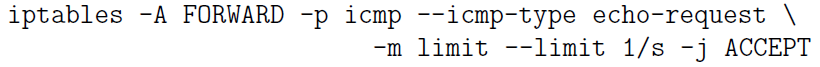
* mac module: used for matching an incoming packets source Ethernet (MAC) address. This only works for packets traversing the PREROUTING and INPUT chains. It provides only one option --mac-source as in “iptables -A INPUT -m mac --mac-source 00:60:08:91:CC:B7 ACCEPT”or “iptables-A INPUT -m mac --mac-ource ! 00:60:08:91:CC:B7 DROP”The second rule will drop all incoming packets unless they are from the specific machine with the MAC address shown.
* limit module: useful in warding off Denial of Service (DoS) attacks.
* SYN-flood protection: The following rule will limit a request for a new connection to one a second. Therefore, if DoS attack consists of bombarding your machine with SYN packets, this will get rid of most of them.



* furtive port scanne: protection against indiscriminate and nonstop scanning of the ports on your machine.



* + protection against ping of death where someone tries to ping your machine in a nonstop fashion



▓port forwarding:

◆when a HTTP request comes in from the internet, it will be received on port 80 that is assigned to HTTP in /etc/services. The firewall would need to forward this request to the LAN machine that is actually hosting the web server.

* by adding a rule to the PREROUTING chain of the nat table, the jump target DNAT stands for Destination Network Address Translation.



* If multiple LAN machines are simultaneously hosting the same HTTP server for reasons of high traffic to the server, you can spread the load of the service by providing a range of addresses for the to-destination option,



▓LOG as a target: if you did not want to drop a packet for some reason, you could go ahead and accept it but at the same time log it to decide later if your current rule for such packets is a good rule.



▓save your firewall rules with iptables-save command: iptables-save > MyFirewall.bk

▓when you reboot the machine, restore the firewall by iptables-restore command: iptables-restore < MyFirewall.bk

▓The -p tcp and -p udp options load into the kernel the TCP and UDP extension modules.

▓When a packet matches a rule whose target is a user-defined chain, the packet begins traversing the rules in that user-defined chain. If that chain doesn’t decide the fate of the packet, then once traversal on that chain has finished, traversal resumes on the next rule in the current chain. Even if the condition part of a rule is matched, if the rule does not

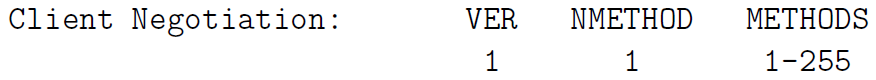
specify a target, the next rule will be considered.

▓firewalls can be designed to operate at any of the following three layers in the TCP/IP protocol stack, often coexist for enhanced security:

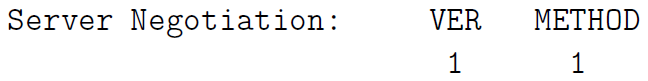
* Transport Layer: packet filtering. Examine every packet, check its IP headers and its higher-level protocol headers (in order to figure out, say, whether it is a TCP packet, a UDP packet, an ICMP packet, etc.) to decide whether or not to let the packet through and to determine whether or not to change any of the header fields.
* Application Layer: HTTP Proxy. examines the requested session for whether they should be allowed or disallowed based on where the session requests are coming from and the purpose of the requested sessions. Such firewalls are built with the help of proxy servers. You need a separate firewall for each different type of service, HTTP, FTP, etc. Such firewalls are basically access control declarations built into the applications themselves. As a network admin, you enter such declarations in the server config files of the applications.
* layer between the Application Layer and the Transport Layer: SOCKS proxy
  + shim layer: using a protocol between the application layer and the transport layer to trap the application-level calls from intranet clients for connection to the servers in the internet
  + intranet: same as LAN (Local Area Network), a local network of computers connected to the rest of the internet through a gateway machine (a router).
  + Using a shim layer protocol, a proxy server can monitor all session requests that are routed through it in an application-independent manner to check requested sessions for their legitimacy. Only proxy server, serving as a firewall, would require direct connectivity to the internet and the local intranet can hide behind the proxy server. The computers in the internet at large would not even know about the existence of your machine in the local intranet behind the firewall.
  + Anonymizing proxy: Proxy servers in general, both at the application layer and at the shim layer, can easily be programmed to give anonymity to the clients who reach out to the service providers in wider internet through such proxies.
  + a proxy server operating at the application layer or the shim layer can carry out data caching (this is particularly true of HTTP proxy servers) that can significantly enhance the speed at which the clients download information from the servers. If the gateway machine contains a current copy of the resource requested, in general it would be faster for a client to download that copy instead of the version sitting at the remote host.

▓SOCKS is referred to as a generic proxy protocol for TCP/IP based network applications, consists of two components: A SOCKS client and a SOCKS server.

* socks client that is implemented between the application layer and the transport layer
* socks server is implemented at the application layer.
* Socksifying the client call: The socks client wraps all the network-related system calls made by a host with its own socket calls so that the hosts network calls get sent to the socks server at a designated port, usually 1080.
* socks server checks the session request made by the socksified LAN client for its legitimacy and then forwards the request to the server on the internet. Any response received back from the server is forwarded back to the LAN client.
* proxy server accepts session requests from SOCKS clients in LAN on a designated port. If a request does not violate any security policies programmed into proxy server, the proxy server forwards request to the internet. Otherwise the request is blocked.
* port forwarding/ tunneling: a proxy server to receive its incoming LAN-side requests for different types of services on a single port and to then forward the requests onwards into the internet to specific ports on specific internet hosts
* proxy server replaces the source IP address in the connection requests coming from the LAN side with with its own IP address. So the servers on the internet side cannot see the actual IP addresses of the LAN hosts making the connection requests. This ploy can maintain complete anonymity with respect to the internet and frequently used by business organizations to hide the internal details of their intranets.
* routing all network traffic through a proxy server as described above also makes it easy to centrally log all internet bound traffic and the caching of web services.
* Compare to nat table, with a SOCKS proxy is that your IP address would remain hidden even when it is a public address
* SOCKS5 has more functions than SOCKS4
  1. built-in support for a variety of client-serve authentication methods
  2. support for UDP and serve as a UDP proxy
  3. move DNS name resolution to the proxy server that, if necessary, can access a remote DNS server. (with SOCKS4, the clients are required to resolve directly the IP addresses of the remote hosts, meaning to carry out a DNS lookup for the remote hosts)
* Interaction Between a SOCKS Client and a SOCKS Server, client wants to access an HTTP server in the internet.:
  1. authentication step: between an SSH client and an SSH server, server needs to authenticate client.
     + socks client opens a TCP connection with the socks server on the servers port 1080.
     + client sends a Client Negotiation packet suggesting a set of different authentication methods that the server could use vis-a-vis the client.



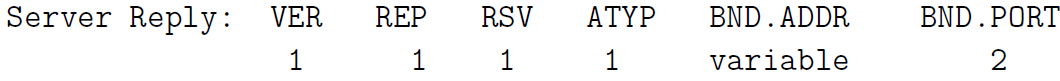
* + - * VER: the version number (SOCKS4 or SOCKS5)
      * NMETHOD: the number of methods that will be listed subsequently for client-server authentication
      * a listing of those methods by their ID numbers, with each ID number as a one-byte integer value.
    - If the socks proxy server accepts the client packet, it responds back with a two-byte Server Negotiation packet:



* + - * METHOD field is authentication method that server wishes to use.
      * socks server then proceeds to authenticate the LAN client using the specified method.
  1. socks client then sends the socks proxy server a Client Request message stating what service it wants at what address in the internet and at which port.



* + - CMD field contains one of three possiblevalues:
      * 0x01 for CONNECT,
      * 0x02 for BIND,
      * 0x03 for UDP Associate.
    - The client always sends a CONNECT request to the socks proxy server after the client-server authentication is complete. For services such as FTP, a CONNECT request is followed by a BIND request.
  1. After receiving the Client Request packet, proxy server evaluates request taking into account the address of the client on the LAN side, the target of the remote host on the internet side and other access control rules typical of firewalls.
     + If the client is not allowed the type of access it has requested, the proxy server drops the connection to the client.
     + If the client is allowed the type of access it has requested, the proxy server sends one or two Server Replies to the socks client. (two replies for BIND requests and one reply for requests CONNECT and UDP)



* BND.ADDR: is the internet-side IP address of the socks proxy server that the remote server will communicate with
* BND.PORT: is the port on the proxy server machine that the remote server sends the information to.
* REP: can take one of the following ten different values:
  + 0x00: successful connection with the remote server
  + 0x01: SOCKS proxy error
  + 0x02: connection disallowed by the remote server
  + 0x03: network not accessible
  + 0x04: remote host not accessible
  + 0x05: connection request with remote host refused
  + 0x06: timeout (TTL expired)
  + 0x07: SOCKS command not supported
  + 0x08: address type not supported
  + 0x09 through 0xFF: not defined
  1. If the connection between the proxy server and the remote server is successful, the proxy server forwards all the data received from the remote server to the socks client and vice versa for the duration of the session.
     + circuit-level proxy: since socks works independently of application-level protocols, it can easily accommodate applications that use encryption. As far as socks server is concerned, there is no difference between an HTTP session and an HTTPS session since, after establishing a connection, a socks proxy server doesn’t care about the nature of the data that shuttles back and forth between a client and the remote host in the internet
* Socksifying a Client-Side Application: Turning a client-side application (such as a web browser, an email client, and so on) into a socks client.
  + By setting the environment variable LD\_PRELOAD, socksify saves you from the trouble of having to recompile your client application so as to redirect the system networking calls to the proxy server.
  + install of Dante to implement a socks proxy server
    - server configuration file /etc/danted.conf
      * Server Settings
        + logoutput: Where the log messages should be sent to.
        + internal: The IP address associated with the proxy server (I chose 127.0.0.1) and the port it will monitor (1080 by default). What is needed is the IP address of the host on which the proxy server is running.
        + external: IP address that all outgoing connections from server

IP address of the interface on which the proxy server will be communicating with rest of the internet

directly name the interface (such as eth0) that the proxy server will use for all outgoing connections

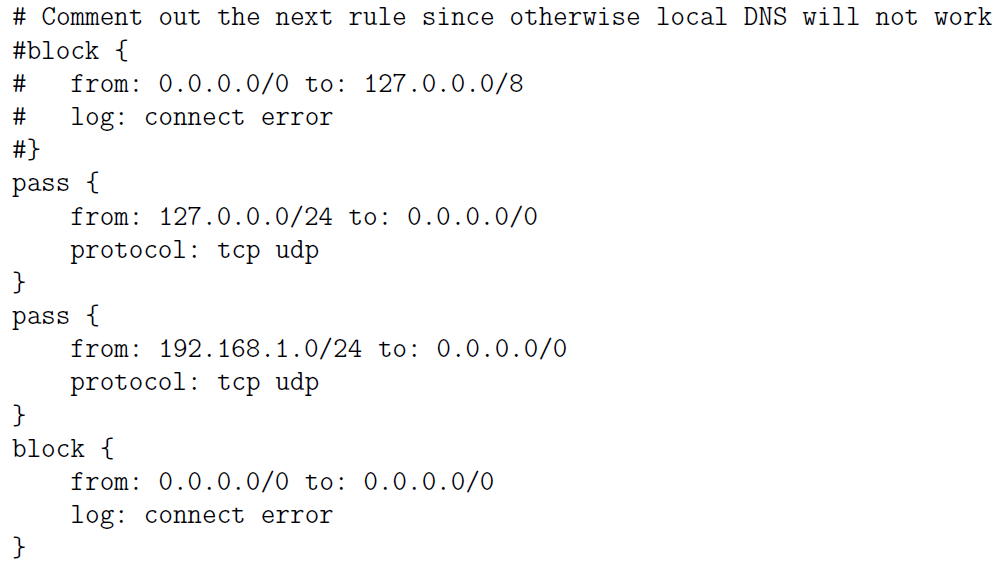
use a socks proxy behind a home or a small-business router wont gain any anonymity from the outgoing IP address used by the SOCKS server since the router will translate the outgoing (the source) IP address into what is assigned to router by the ISP anyway.

* + - * + method: Methods are for authenticating the proxy clients.
        + user.privileged: If client authentication requires that some other programs be run, the system would need to run them with certain specified privileges. For that purpose, you can create a user named proxy if you wish and set this option accordingly.
        + user.notprivileged: This specifies as to what read/write/execute privileges the server should be set to when running in the default non-privileged mode. Set it to which means that the server would have no nobody permissions at all with respect all the other files in the system.
      * Rules
        + client rules: rules that control as to which socks clients are allowed to talk to the proxy server.



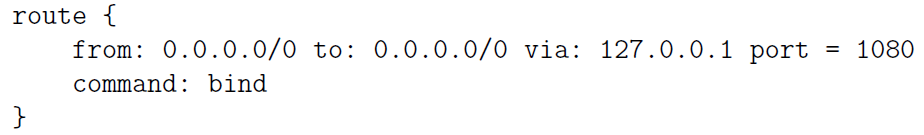
These rules say to allow all local socks clients on the same machine and all socks clients on the local LAN to talk to the SOCK proxy server on this machine. The third rule says to deny access to all other socks clients.

* + - * + rules that control as to what remote services the proxy server can be asked

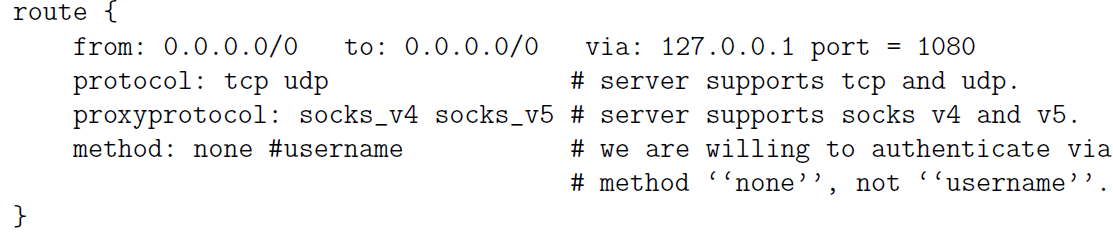


any local socks client will be able to call on any service anywhere for a TCP or UDP service. The third rule does the same for any socks client in the local LAN. The fourth rule blocks all other socks client requested services.

* + - * Routes
    - client configuration file /etc/dante.conf: regulates the behavior of a socksified client
      * The bind command allows incoming connections for protocols like FTP in which the local client first makes a control connection with a remote server and the remote server then makes a separate connection with the client for data transfer



* The following rule tells the client where the SOCKS proxy server is located and what port the server will be monitoring, the server supports TCP and UDP services, both SOCKS4 and SOCKS5 protocols, and that the server does not need any authentication



▓elite / high-anonymity proxy server: an HTTP proxy server does not send any of these fields (HTTP\_X\_FORWARDED\_FOR, HTTP\_VIA, HTTP\_PROXY\_CONNECTION) to the remote server.

▓Squid can also be used as an anonymizing proxy server to control access to web (HTTP,

HTTPS) and FTP resources

* Usage
  + Through combination of ACL (Access Control List) and http access directives in config file to control who is allowed to connect with proxy from what IP.
  + To designate a list of domains that would remain blocked for all access through the proxy.
  + designate the times during the day, or even a week, when the proxy should allow certain types of access, with the access remaining closed at other times.
* Web caching: if you make repeated requests to the same web page and there exists a web proxy server between you and the source of the web page, the proxy server will send a quick request to the source to find out if the web page was changed since it was last cached. If not, the proxy server will send out the cached page.
  + parent-child: If a child cache cannot find an object, it passes on request to parent cache. If the parent cache itself does not have the object, it fetches and caches object and then passes it to on to the child cache that made the original request.
  + Sibling: useful for load distribution. Before a query goes to the parent cache, the query is sent to adjacent sibling caches.
* Since Squid is a caching proxy server, it must avoid returning to the clients objects that are out of date. So it automatically expires such objects. You can set the refresh time in the configuration file to control how quickly objects are expired.
* Squid has the notion of Safe ports, these are ports that Squid will forward. If an incoming request received by proxy server is for one of these ports at destination server, it will let that request through. If the port requested at the destination port is outside of the designated Safe ports, they would not go through the server.

▓Harvest broker first constructs the index (gathers information from designated sources that may be reside on your own hard disk, then creates an efficiently searchable index for the gathered information) and then serves it out through a web interface.

▓Information Security: tegrity and/or confidentiality of information from sending

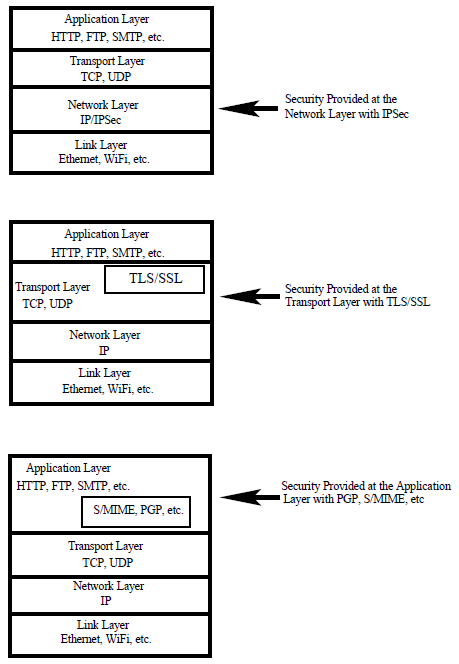
point to the receiving point in a network and sender is actually the entity that the information was received from.

▓from end-to-end perspective, ensuring information security in network-centric applications requires paying attention to

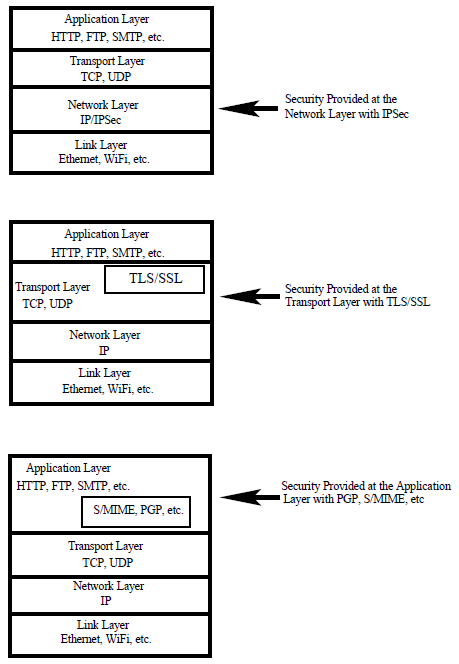
* Authentication:
  + the source is indeed as alleged in the information.
  + data integrity: The information was not altered along the way.
* Confidentiality: information is safe from being eavesdropped on during its transit from the sending point to the receiving point.
* Choosing the best security parameters and key management: two endpoints of a communication link may possess different computational capabilities and not have access to the same security algorithms. Key management is providing solutions to the practical problems that arise when users possess multiple public/private key pairs.

▓end-to-end information security may be provided at different layers in the internet suite of communication protocols

* security services in the Network Layer by using the IPSec protocol: if solely relied upon, makes it difficult to customize the security policies to specific applications. It also takes away the management of security from the application developer.



* security in a higher layer by adding security related features to TCP packets. This can be done with a Session Layer protocol like the Secure Sockets Layer (SSL/TLS).【security provided by iptables also operates at the transport layer. However, that is primarily defensive security. Iptables based firewall security is not meant for information security through authentication and confidentiality.】
* embed security in the application itself: applications PGP, S/MIME, … are security aware. 【proxy servers can also provide security at the application level. However, with iptables, that is again primarily defensive security in the form of access control. It is generally not the job of the proxy servers to provide security authentication and confidentiality services for information security】

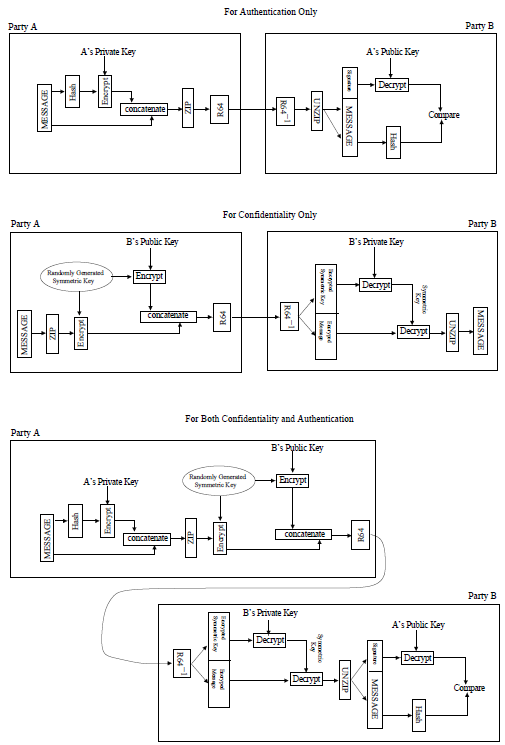
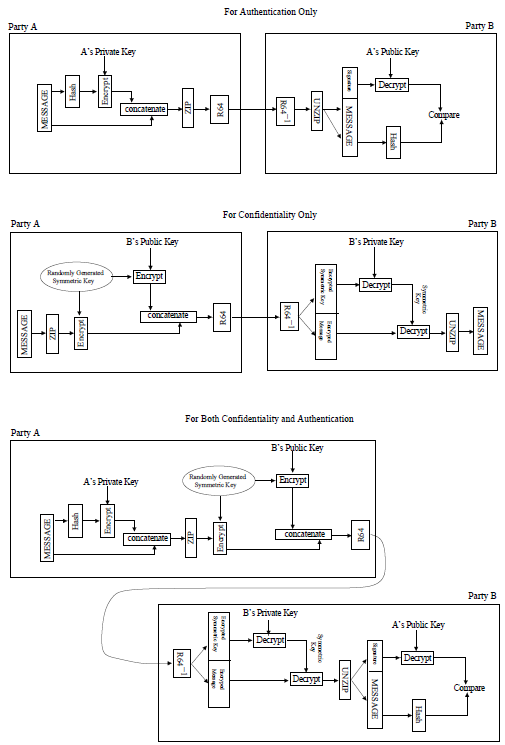


* In each of the three different layers mentioned above
  + authentication can be provided by public-key cryptography and key management issues in all layers can be made complicated by the fact that, in general, users are allowed to have multiple public keys.
  + confidentiality can be provided by symmetric key cryptography

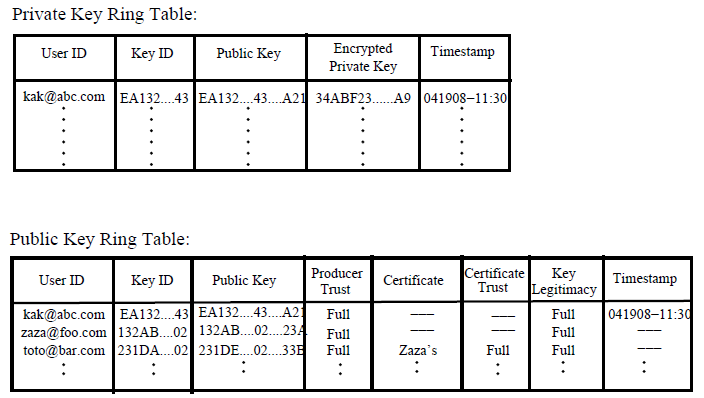
▓internet standards often use octet for a byte and not infrequently datagram for a packet.

▓PGP (Pretty Good Privacy): What makes PGP particularly important is that it is now widely used for protecting data in long-term storage.

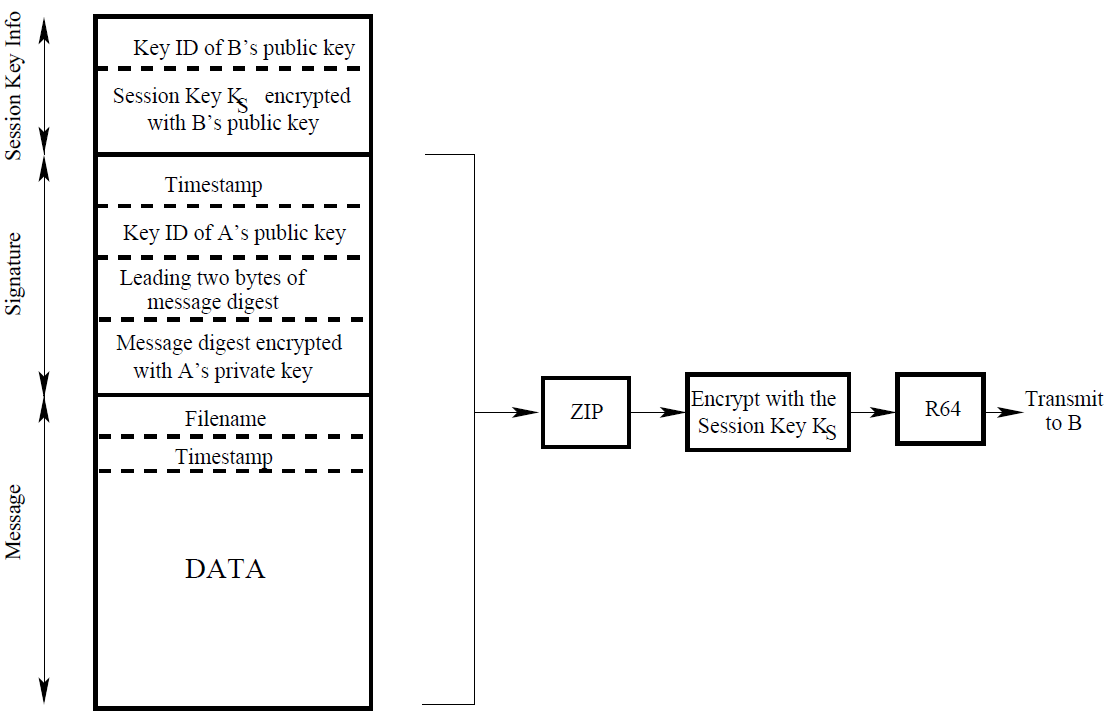
* PGPs operation consists of five services:
  + Authentication Service: sender attaching his/her digital signature to the email and the receiver verifying the signature using public-key cryptography.
    - RSA/SHA based digital signature
      1. At the senders end, a SHA hash function is used to create a 160-bit message digest of the outgoing email message.
      2. message digest is encrypted with RSA using the senders private key and the result prepended to the message. The composite message is transmitted to the recipient.
      3. receiver uses RSA with senders public key to decrypt the message digest.
      4. receiver compares locally computed message digest with received message digest.
    - DSS/SHA based signatures: DSS is Standard Digital Signature
    - PGP also supports detached signatures that can be sent separately to the receiver. Detached signatures are also useful when a document must be signed by multiple individuals.
  + Confidentiality Service: PGP uses symmetric-key encryption for confidentiality. The user has a choice of three different block-cipher algorithms for this purpose: CAST-128, IDEA, or 3DES
    - block ciphers are used in the Cipher FeedbackMode (CFB)
    - session key (128-bit encryption key) is generated for each email message.
    - session key is encrypted using RSA with the receivers public key. Session key can also be established using the ElGamal algorithm.
    - What is put on the wire is the email message after it is encrypted first with the session key and then with the receivers public key.
    - If confidentiality and sender-authentication are needed, digital signature for the message is generated using the hash code of the message plaintext and appended to the email message before it is encrypted with the session key.
  + Compression Service: PGP compresses email message after appending signature but before encryption. This makes long-term storage of messages and their signatures efficient. This also decouples encryption algorithm from the message verification procedures. Compression is carried out with the ZIP algorithm.
  + E-Mail Compatibility Service: Since encryption, even when it is limited to the signature, results in arbitrary binary strings, and since network message transmission is character oriented, we must represent binary data with ASCII strings. PGP uses Base64 encoding for this purpose.
  + Segmentation Service: For long email messages (messages with attachments), many email systems place restrictions on how much of the message will be transmitted as a unit. For example, some email systems segment long email messages into 50000 byte segments and transmit each segment separately. PGP has built-in facilities for such segmentation and re-assembly.
* the three different modes in which PGP can be used for secure email exchange.



* Key Management Issues in PGP
  + public key encryption is central to PGP. For authentication, sender uses his/her private key for placing his/her digital signature on the outgoing message. For confidentiality, sender uses the receivers public key for encrypting the symmetric key used for content encryption.
  + We can expect people to have multiple public and private keys. For example, an individual may wish to retire an old public key but allow for a smooth transition, may make available both the old and the new public keys for a while. A PGP message consists of three components: a session key component, a signature component, and the actual email message itself. The leading two bytes of message digest determines the correct public key (of the sender) was used to decrypt the message digest for authentication. These two bytes also serve as a 16-bit for the actual email message. The message frame check sequence digest itself is calculated using SHA-1.
  + PGP must allow the receiver have stored multiple public keys for a given sender. Problem is that it is wasteful in space because RSA public keys can be hundreds of decimal digits long. PGP protocol solves this problem by
    - using key ID (relatively short key identifiers): key ID associated with a public key consists of its least significant 64 bits, so always just 8 bytes long. Each hex string begins with the least significant byte. Therefore, the sixteen hex characters in a key ID will always be the same as the first sixteen hex characters of the public key.
    - requiring every PGP agent maintain its own list of paired private/public keys in the Private Key Ring; and a list of the public keys for all its email correspondents in the Public Key Ring.
    - keys for a particular user are uniquely identifiable through a combination of the user ID and the key ID.



* PGP stores the private keys in the table in an encrypted form so that the keys are only accessible to the user who owns them.
* fields Producer Trust, Key Legitimacy, Certificate, Certificate Trust are to assess how much trust to place in public keys belonging to other people.
* The entry stored in the field is where the public key Public Key is stored.
* Entry in the Producer Trust field of the Public Key Ring table indicates the extent to which the owner of a particular public key can be trusted to sign other certificates. Generally be one of three values: full, partial, or none.
* Entry in Certificate field holds the certificate(s) that authenticates the entry in the public key field. The third row in the Public Key Ring in Figure shows that toto public key was signed by zaza. That is, zaza supplied the certificate that authenticated totos public key, zaza used its private key to digitally sign toto public key and sent that signed document to kak.
* Certificate Trust field indicates how much trust a user wants to place in the entry in the Certificate field.
* For a given public key, the value for the field Key Legitimacy is automatically derived by PGP from the value(s) stored for the Certificate Trust field(s) and a predefined weight for each symbolic value for certificate trust. An individual may receive multiple signed certificates for a new potential email correspondent from others in a web of trust.
* Party A sends a PGP message to party B

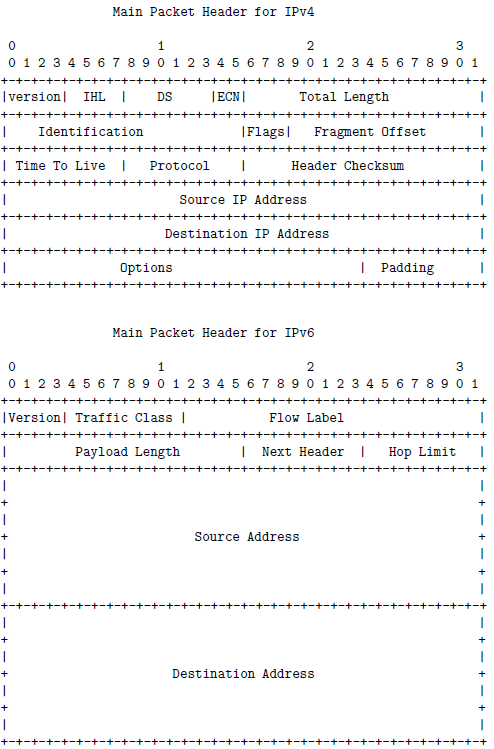


* PGP’s Web of Trust: A unique feature of PGP is its own notion of a certificate authority for authenticating the binding between a public key and its owner. Web of trust is a bottom-up approach to establishing trust for authentication.
  + users public key can be signed by any other user, everyone who signs a public key for another submits the signature to a central key server. For example, in kaks public key ring, totos public-key was signed by zaza. Kak fully trusts zaza because zaza handed its public key to kak directly (say, over the phone). Because the fully-trusted zaza endorses the new user totos public key, toto also becomes a fully-trusted email correspondent for the user kak.
  + Because there is no hierarchy of trust in PGP, it is possible that a user will receive two different certificates for a new email correspondent, say one that the receiver will trust fully and the other that the receiver may trust only partially. Whether or not to trust such a potential email correspondent is up to the receiver of the certificates.
* creation of the web of trust is facilitated by the availability of free publicly available PGP Key-servers at various places around the world. In order to upload your key to such a server, one typically creates a GPG (Gnu Privacy Guard) key though the following steps:
  + create a new .gnupg directory at the top level of your home directory.
  + execute the key generation com gpg mand to create a public/private key pair: gpg --gen-key
    - you will supply type of keys, when the key should expire, User-ID to identify key, email address.
    - If the entropy found is insufficient to create a true random number, you will be asked to make mouse movements and random keyboard entries for increasing the entropy.
    - Gpg will output a 40-character Key Fingerprint and KeyID consists of the last 8 characters of the Key Fingerprint.
    - public and private keys that are generated are deposited in the files pubring.gpg and secring.gpg of the .gnupg directory.
    - Generated trustdb.gpg. file that keeps the trust database
  + export your public key to one of the worldwide PGP key servers by

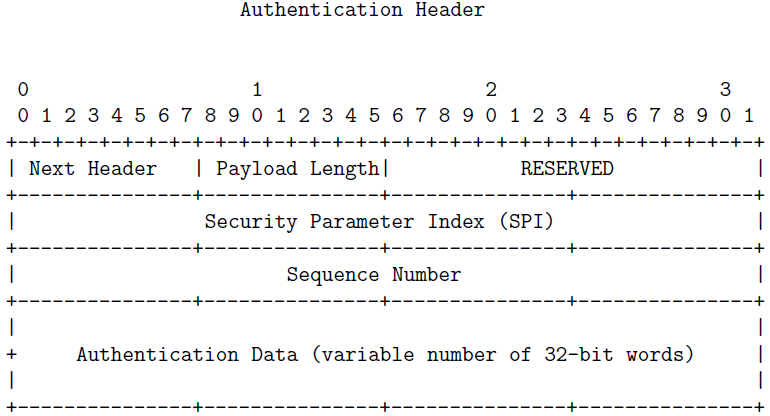
gpg --keyserver pgp.mit.edu --send-keys your\_8\_char\_KeyID

▓IPSec: providing authentication, confidentiality, and key management at the level of IP packets in the Network Layer of the TCP/IP protocol stack

* When security is implemented at the Network Layer in the TCP/IP protocol, it covers all applications running over the network. That makes it unnecessary to provide security separately for, say, email exchange, running distributed databases, file transfer, remote site administration, etc.
* The largest application of IPSec is in Virtual Private Networks (VPN). A VPN is an overlay network that allows a group of hosts that may be widely scattered in the internet to act as if they were in a single LAN.
* IP-level authentication is provided by inserting an Authentication Header (AH) into the packets: means that the source of the packet is as stated in the packet header and that the packet was not altered during transmission. AH stores a hash value for those portions of a packet that are expected to stay invariant during its transmission
* IP-level confidentiality is provided by inserting an Encapsulating Security Payload (ESP) header into the packets.
* IPSec is used in two different modes:
  + Transport Mode: regular mode for packets to travel from a source to its destination in a network, except for that the two endpoints must carry out the security checks on the packets on the basis of the information contained in the authentication header.
  + Tunnel Mode: source and destination endpoints for a given packet stream may not have the ability or the resources to carry out the security checks on packets.
    - So a source must route the packets to a designated location P in the network for inserting the authentication and/or ESP headers.
    - If the originally intended destination also is not able to carry out the security checks on the packets, P may need to send the packets to another designated location Q to carry out the security verification on the basis of the information in the security headers inserted by P and send the packets thus verified to their true destination.
    - Tunnel: Encapsulator (P) and decapsulator (Q)
    - IP-in-IP protocol is used to send packets to the designated point P: encapsulating the IP header that has the actual source and destination fields with another IP header that first sends the packet to a designated location.
* IPSec includes so that only specified traffic filtering capability need be subject to security processing.
* if the two endpoints of a communication link are able to their own authentication processing, you will use IPSec in the Transport Mode with just the additional AH headers. If the endpoints cannot do their own authentication, you will have to use IPSec in the Tunnel Mode.
* if you want to use IPSec for confidentiality, you’ll need to the ESP headers (with or without the AH headers since the ESP headers can also carry out authentication).
* IPSec security features are implemented that follow the main IP as extension headers header in an IP packet.
  + IPv4 header
    - Total Length: 16-bit word, designates the total length of the overall packet (including the data payload) in bytes.
    - Identification, flags, Fragment Offset: hold values that are assigned by the sender to help the receiver with the re-assembly of the IP fragments back into an IP datagram.
    - Time to Live: 8 bits, is subtracted by 1 for each pass through a router.
    - Protocol: field of the IPv4 header plays an important role in grafting IPSec onto IPv4. Ordinarily this field indicates the next higher level protocol in the TCP/IP stack that is responsible for the contents of the data field of the IP packet. When IPSec is used, this field contains the integer value that represents the security header to follow the main header. Integer 50 represents the ESP header that is used for encryption services in IPSec. Therefore, if the next header is the ESP header, number 50 will be stored in the Protocol field. Number 51 represents the AH protocol that is used for authentication services.
  + IPv6 header: using an arbitrary number of headers, headers that follow the main IPv6 header are called the extension headers (Authentication Header and Encapsulating Security Payload Header.)



* + Authentication Header (AH)
    - Payload Length field: specifies the length of the AH in 32-bit word, minus the integer 2.
    - Security Parameter Index (SPI) field: 32-bit, establishes the Security Association (SA), a grouping of the security parameters needed for authentication, involve a public key identifier, an initialization vector identifier, an identifier for the hashing algorithm used, etc., used for authentication. The Security Parameter Index along with the source IP address is used to establish the Security Association of the sending party.
    - Sequence Number field: 32-bit integer, an increasing number for each packet sent to prevent replay attacks. For each SPI, only one packet can have a given sequence number. If an adversary capture some of the IP packets and re-transmit them to the destination, the destination IP engine would detect that there was a problem when it starts receiving multiple packets with the same sequence number for the same value of SPI.
    - Authentication Data Field: holds MAC (Message Authentication Code) of packet calculated with either SHA-1 hash function or HMAC algorithm. MAC is calculated over the IP header fields that do not change in transit including the source and the destination IP addresses, the AH header (but without the Authentication Data), and the inner IP packet for establishing authentication in the tunnel mode. If the receiver compute a different MAC in the header, drop the packet



* + - IPv4 packet without AH header



* + - IPv4 packet with AH header

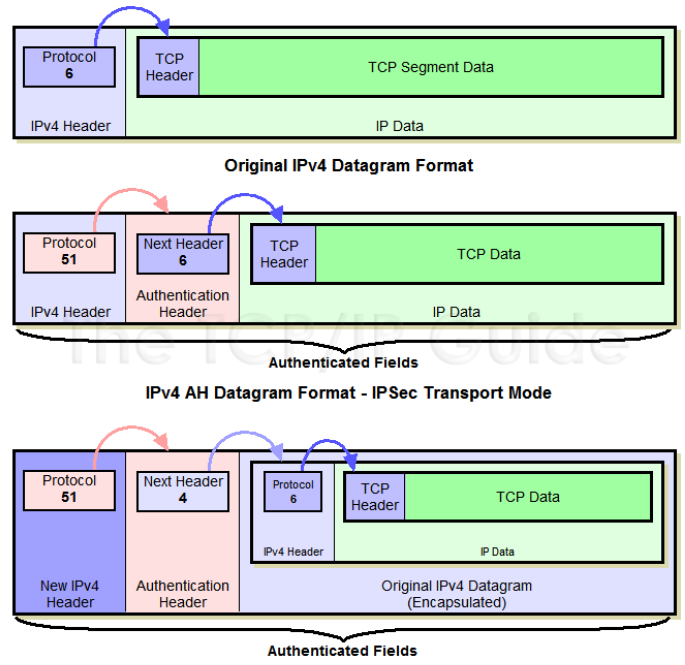


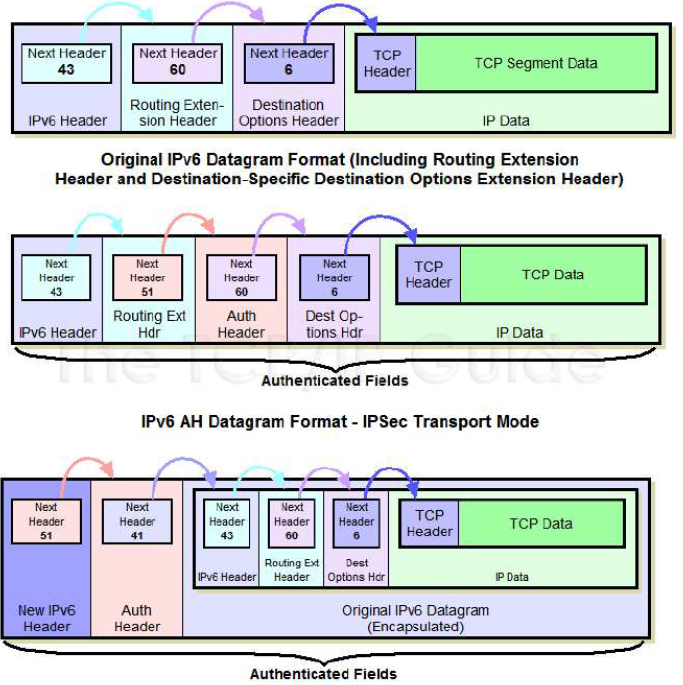
* + - IPv6 packet without AH header



* + - IPv6 packet with AH header







* + Encapsulating Security Payload (ESP) and Its Header39

