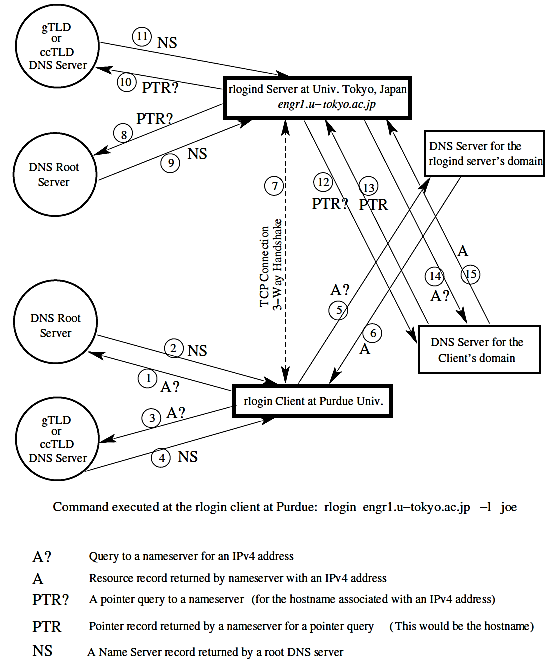
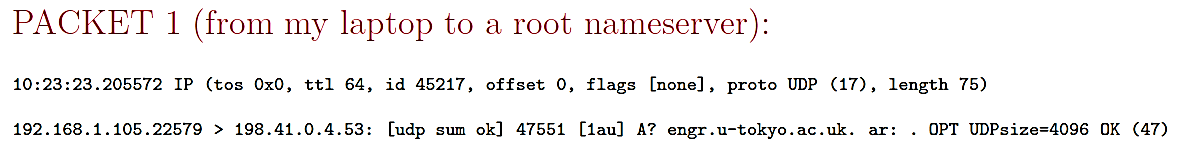
▓DNS(Domain Name Service / Server / System / Space): translate symbolic hostnames into the numerical IP addresses and vice versa. Lists mail exchange servers that accept email for different domains.

* 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 downloaded by clicking on a URL in a spam email may overwrite the entries in the file /etc/resolv.conf and 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 trying to reach, you could end up giving your personal 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.)

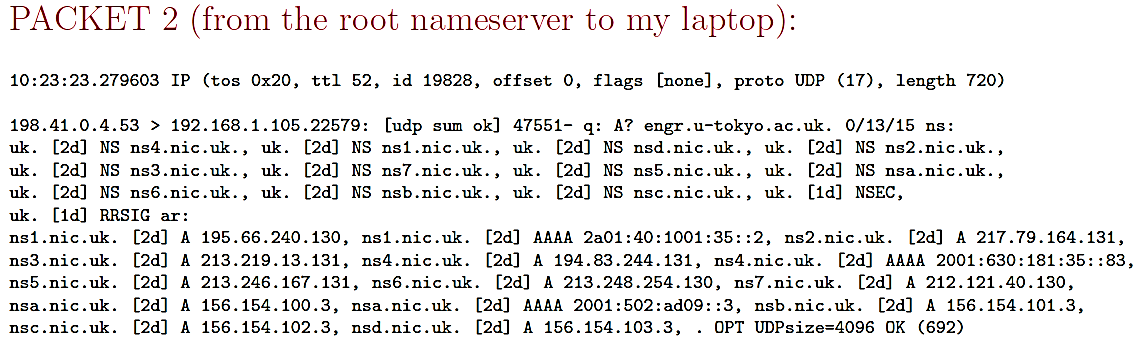


1. on the client side first enters the rlogin command (a Unix program that allows users to log in on another host using a network).
2. client resolver first contacts one of the root nameservers for where to go for resolving the names that end in .jp
3. root nameserver responds back with the IP address 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.

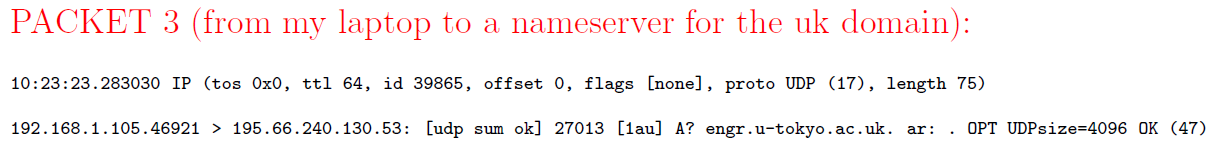
* most commonly a DNS request for name lookup is sent out in a UDP packet.
* UDP packet consists of an 8 byte header:
  + 2 bytes for the source port
  + 2 bytes for the destination port
  + 2 bytes for the length of the packet, includes the length of the header
  + 2 bytes for the checksum.
* Since 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 > 198.41.0.4.53” in the first packet says that my laptop, whose 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.



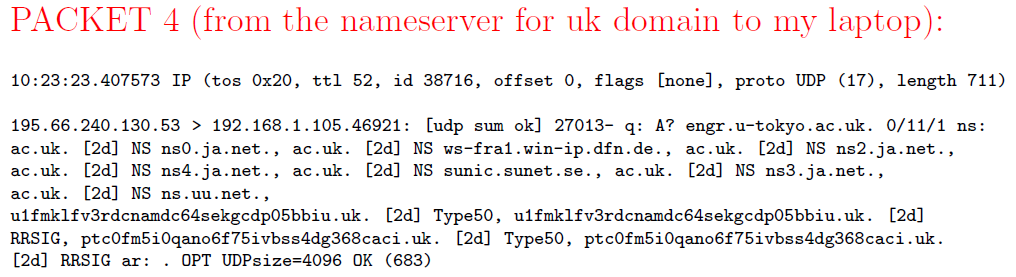
* + The answer returned by the root nameserver consists of the symbolic names and subsequently the IPv4 addresses for several nameservers responsible for the uk domain. For example, one of the nameservers listed for the uk domain is ns1.nic.uk and its IPv4 address is 195.66.240.130
  + A string such 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.

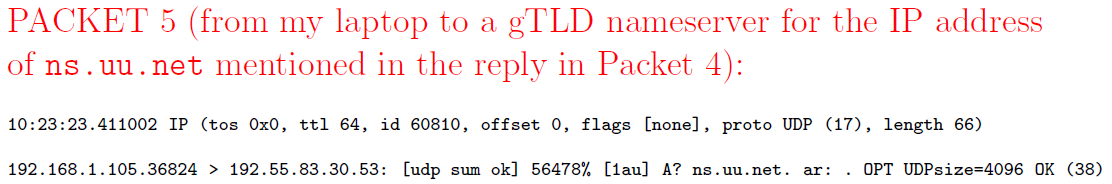


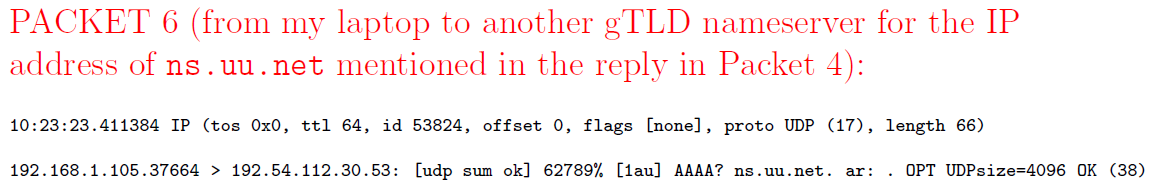
* + String 192.168.1.105.46921 > 195.66.240.130.53 tells us that this is a packet from my laptop to the nameserver for the ns1.nic.uk uk top-level domain. Transaction ID number is 27013.



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

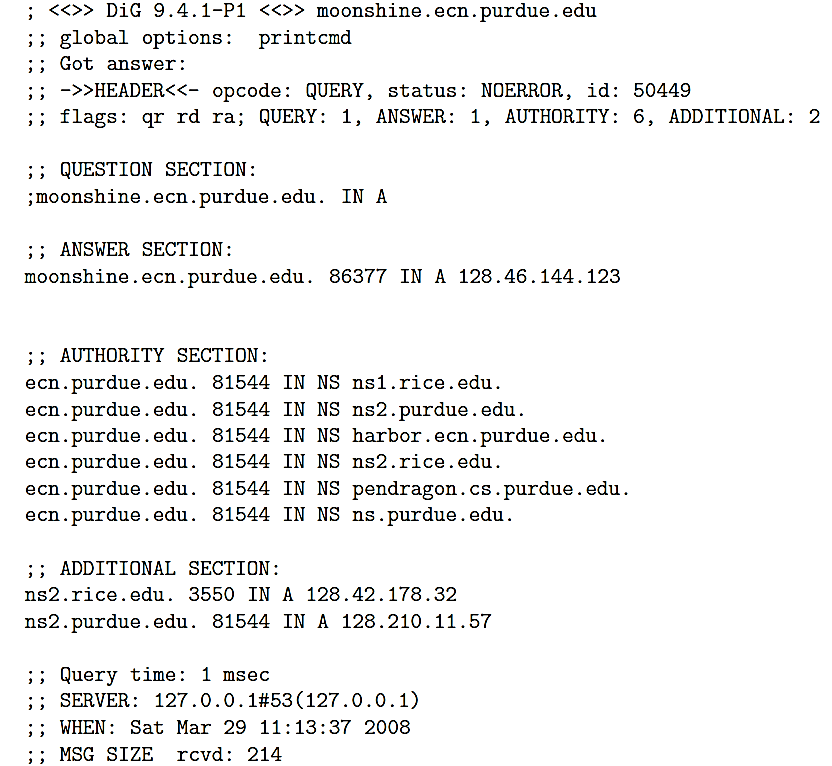




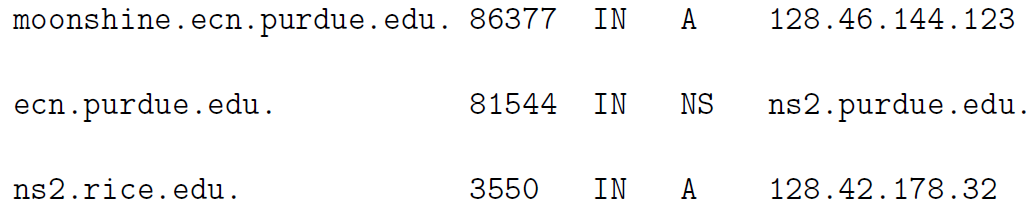


* each host is represented in DNS by two DNS records: an address record and a reverse mapping pointer record.
* fully qualified domain names (FQDN): The root domain is represented by a period ”.” and formal DNS name of the purdue.edu domain is purdue.edu. period at the end stands for the root of the 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

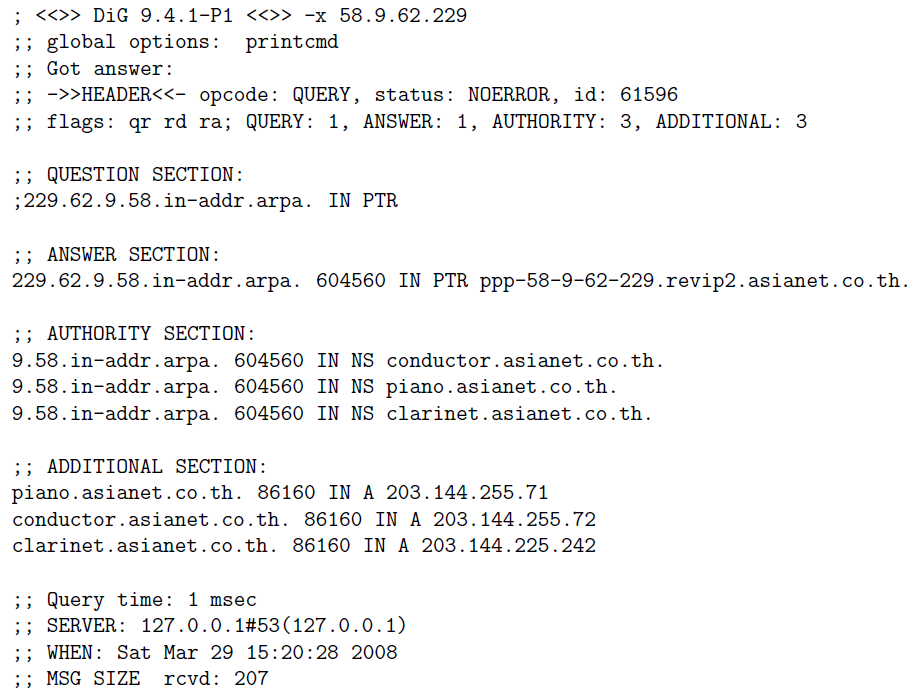
▓dig (domain information groper): is a useful command for interrogating DNS nameservers for information about the host IP addresses, mail exchanges, nameservers for other domains, and so on. Numbers like 86377, 81544, 3550 are TTL in seconds. (EX. dig moonshine.ecn.purdue.edu)



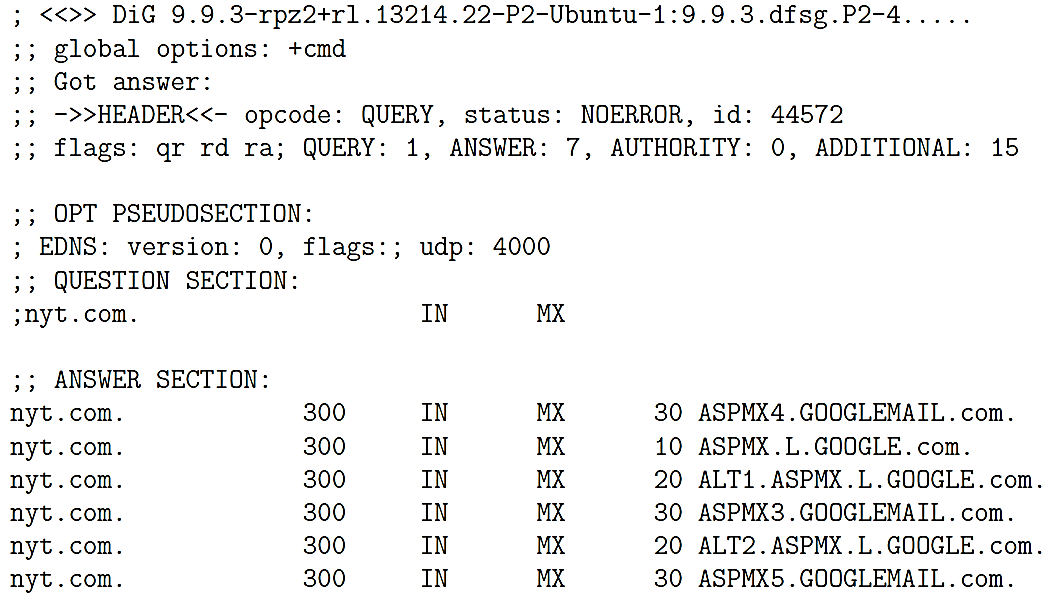
* AUTHORITY SECTION: lists nameservers for the ecn.purdue.edu domain.
* Additional Section: lists the IP addresses of the nameservers named in the Authority Section.
* Resource Record (RR) consists of 5 items:

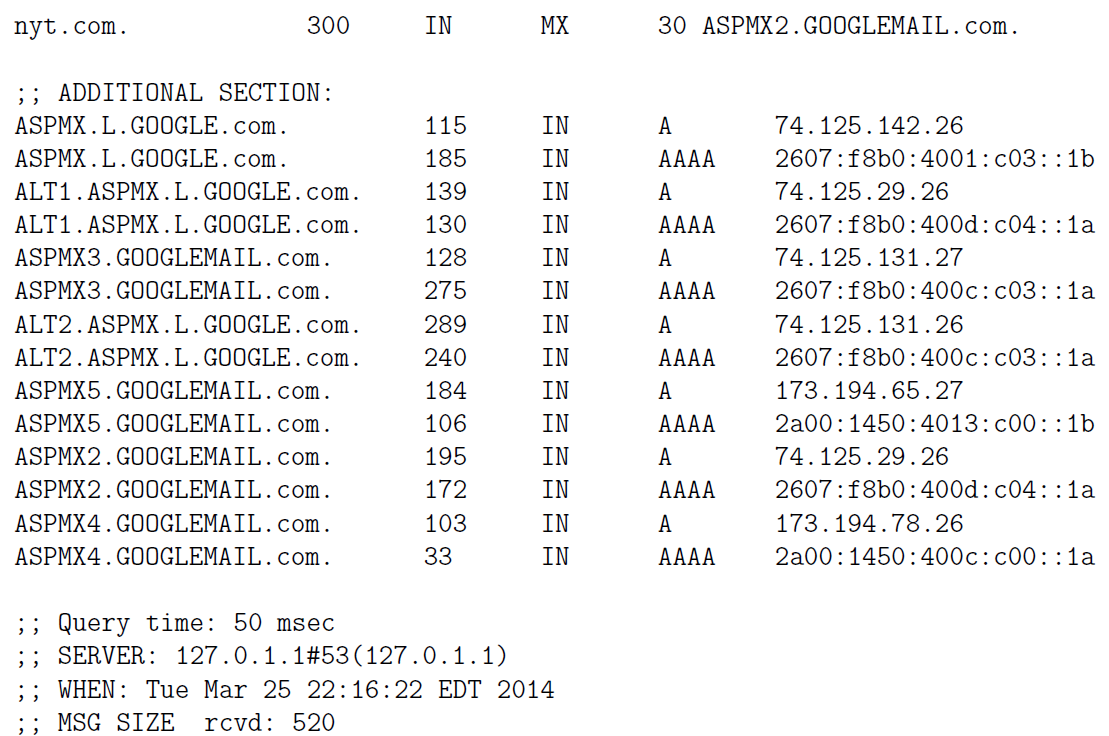


* fully qualified domain name (FQDN): such as ns2.rice.edu.
* Time-to-live (TTL): such as 86377 seconds
* Class of the record: such as IN stands for class internet
* Type of the record.
  + A: address record in the form of an IPv4 numerical address.
  + AAAA: address record in the form of an IPv6 numerical 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.

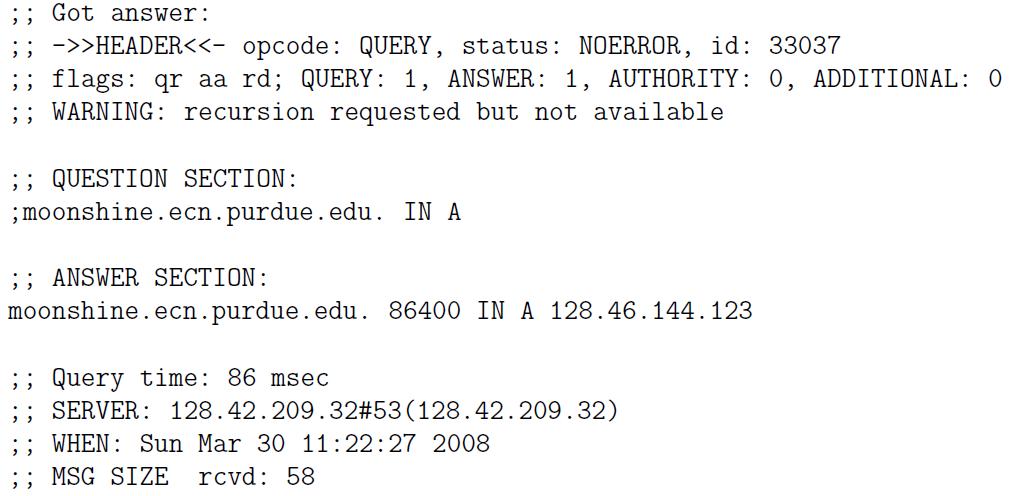


* dig nyt.com MX: the IP address of the host (or hosts) responsible for mail exchange for a domain nyt.com
  + when multiple hosts are returned for the mail exchange service for a domain and each has its own MX preference number, the MX hosts with the smallest preference numbers must be tried first for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



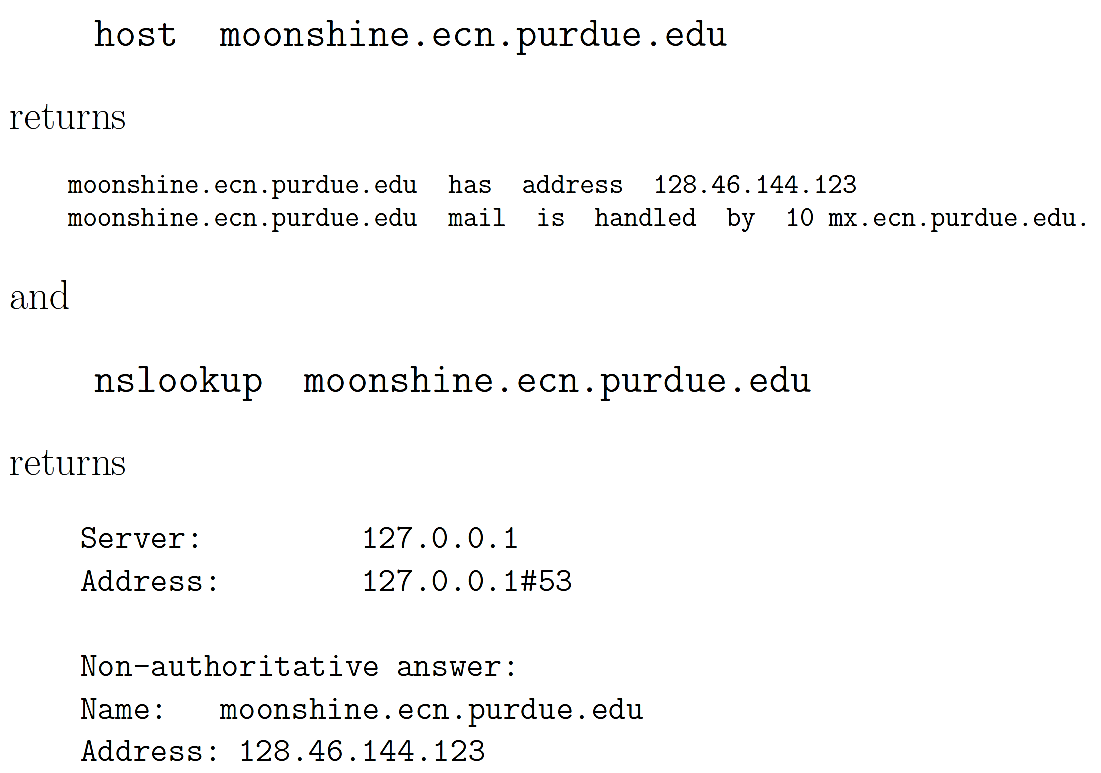


◆dig @ns1.rice.edu +nocmd moonshine.ecn.purdue.edu: ask the DNS server running at Rice University for the IP address for moonshine.ecn.purdue.edu:



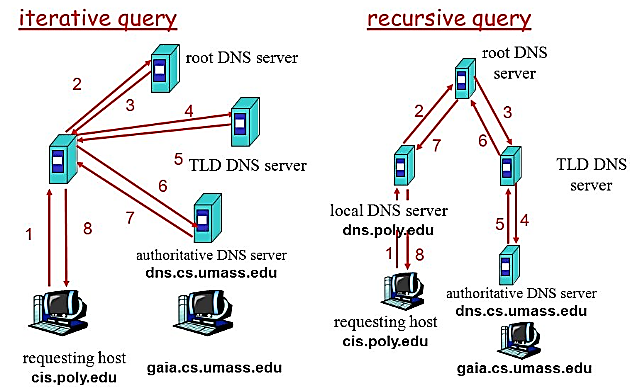
* dig examines the contents of your /etc/resolv.conf file for the nameservers to send the query to the nameservers in the order they are listed in the /etc/resolv.conf file.

▓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 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 name resolution of www.nyt.com, local nameserver would be able to resolve URL by looking into its own cache.
* the various client applications maintain their own DNS caches usually with very short caching times
* operating system may also maintain a local cache for the previously resolved hostnames with relatively 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.

▓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 amazon.com domain.

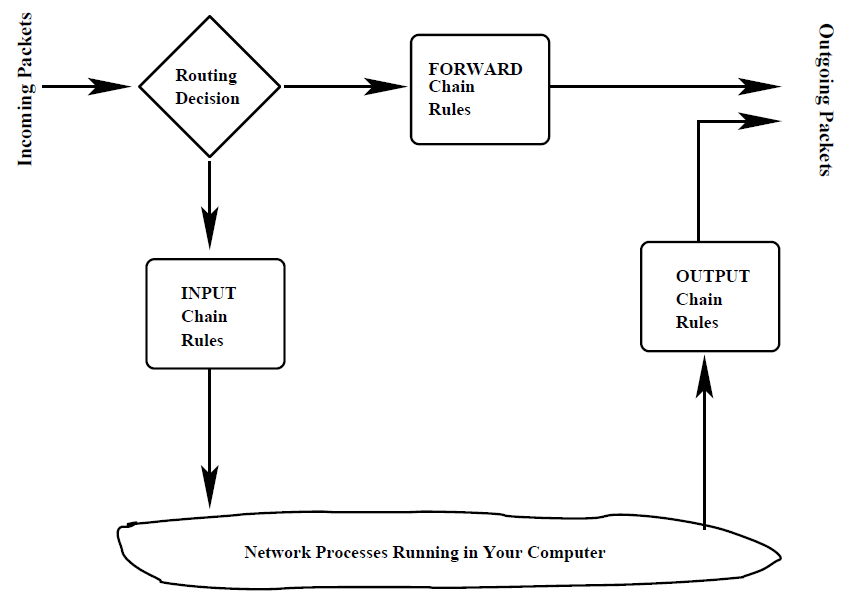
1. asking the DNS server running at harbor.ecn.purdue.edu to carry out the name lookup for the domain amazon.com by dig amazon.com @harbor.ecn.purdue.edu
2. DNS server will make an NS query to nameserver in charge of the com top-level domain for the IP addresses of the nameservers in charge of the amazon.com 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.

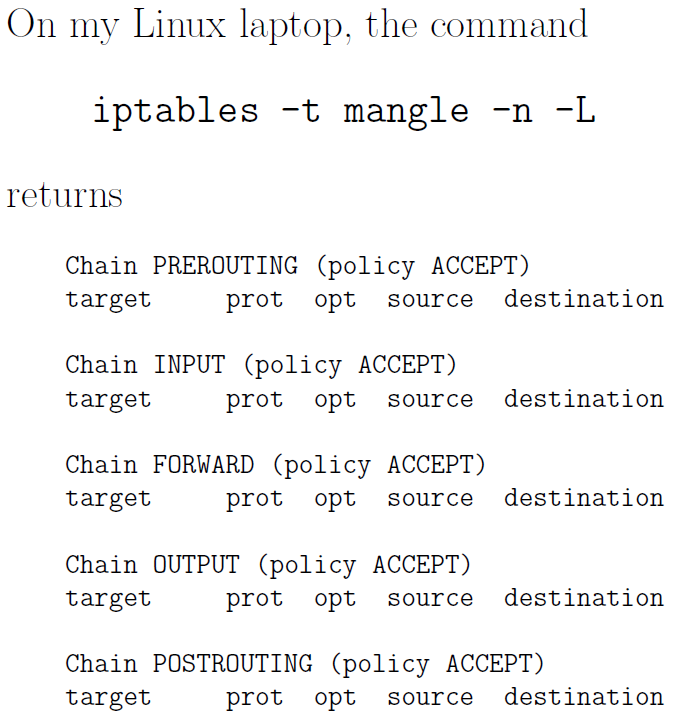
* To place in the cache of the nameserver ns.purdue.edu a fake IP address for [www.foo.com](http://www.foo.com). Attacker query nameserver for purdue.edu domain for possibly nonexistent symbolic hostnames 1.foo.com, 2.foo.com etc.
* The nameserver ns.purdue.edu will have no entries for this hostnames 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

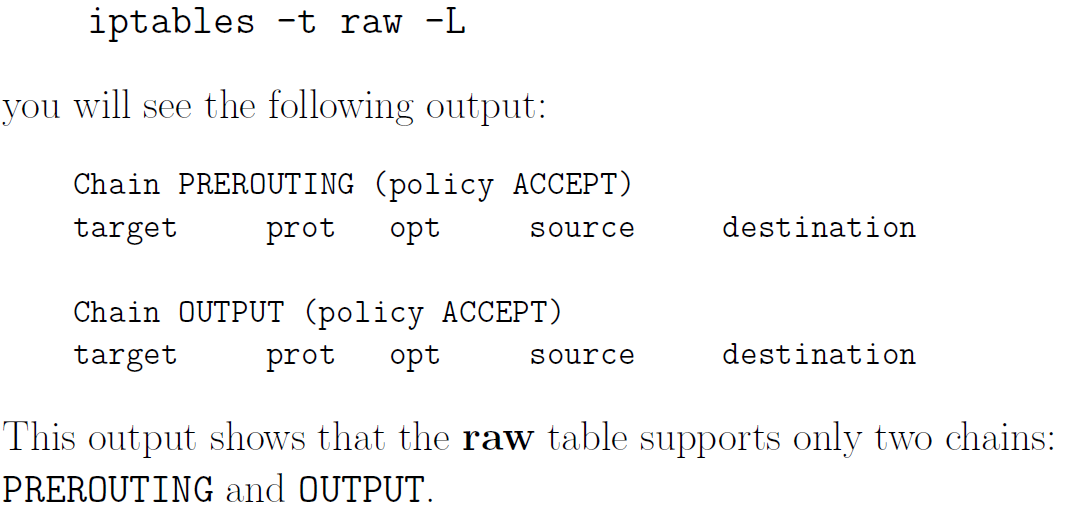
* Two primary types of firewalls
  + packet filtering firewalls
  + proxy-server firewalls: all network traffic meant for the computers in a LAN is routed through a proxy server that allows the proxy server to exercise access control over the traffic
* Linux kernel uses four tables, each consisting of 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. 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 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.
  + 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.
    - 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 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 packet’s source address field or 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 (actions that are permitted for the rules):
     + DNAT: when 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, you 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: change 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: substitute a single previously specified IP address for the source address in the outgoing packets, MASQUERADE can substitute a DHCP IP address 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.

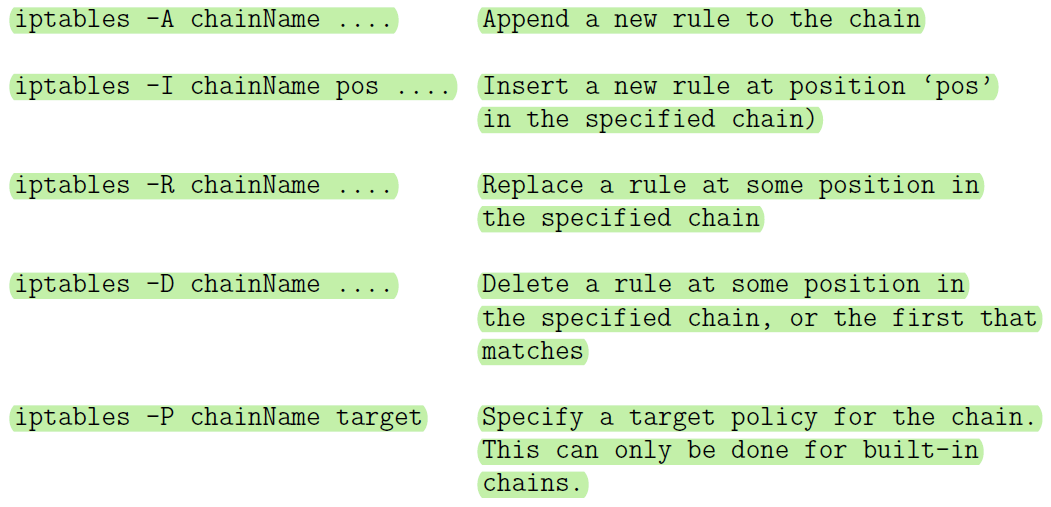
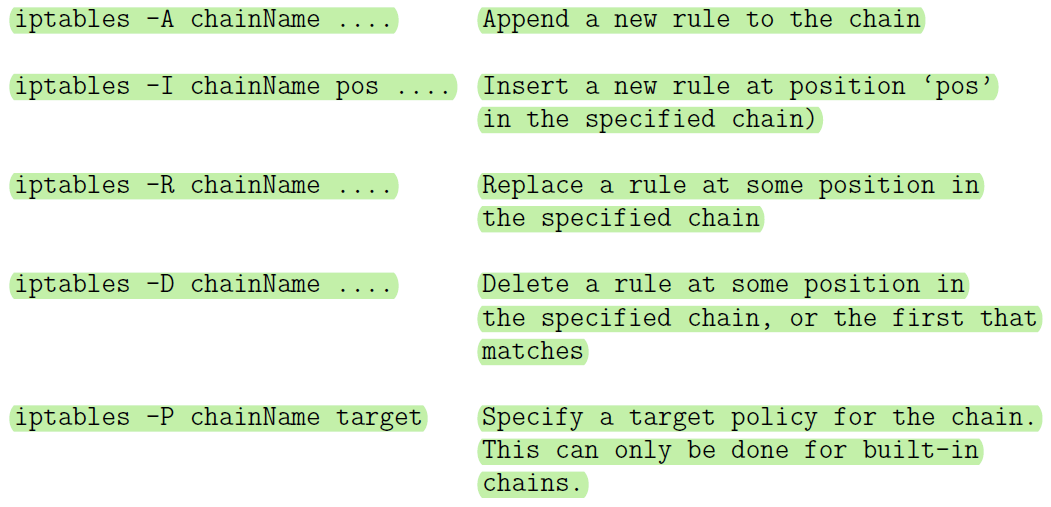


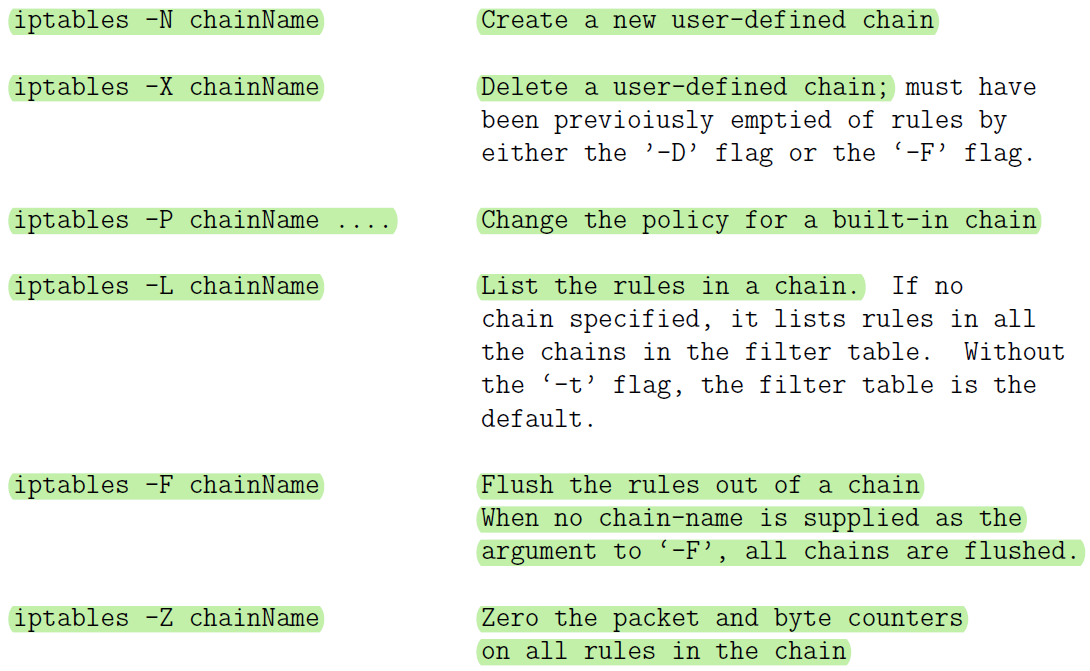
* 1. TOS: change the TOS field in a packet.
  2. TTL: change the TTL field of the packet.
  3. MARK: give a special mark value to the packet. Such marks are recognized by the iproute2 program for routing decisions.
  4. SECMARK: sets up a security-related mark in the packet. Such marks can be used by SELinux fine-grained security processing of the packets.
  5. CONNSECMARK: places a connection-level mark on a packet for security processing.
* STRUCTURE OF THE raw TABLE: specifying the exemptions from connection tracking. When rules are specified for the raw table, the table takes priority over the other tables.



▓iptables are created by the iptables command 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.

* If a packet is the first that the firewall sees or knows about, it is 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 in state ESTABLISHED.
* Connection tracking is also used by the nat table and by its MASQUERADE target in the tables.
* extension modules makes it possible to carry out connection tracking. An extension module is specified by the -m option
* state module: interpret the connection-tracking analysis produced by the ip\_conntrack module.



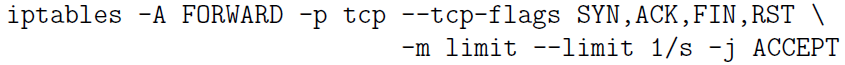
* how the interpretation of the results on a packet produced by the ip\_conntrack module should be carried out is specified by the additional ‘—state’ option.
* --state suboption supplies a comma-separated list of states of the packet that must be found to be true for the rule to apply. These states 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.



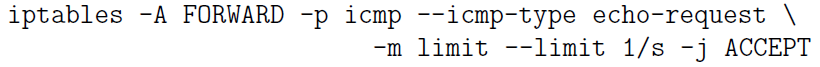
* mac module: matching an incoming packets source Ethernet (MAC) address. This only works for packets traversing the PREROUTING and INPUT chains. It provides only one option --mac-source as in “iptables -A INPUT -m mac --mac-source 00:60:08:91:CC:B7 ACCEPT”or “iptables-A INPUT -m mac --mac-ource ! 00:60:08:91:CC:B7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



* furtive port scanner: 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. 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

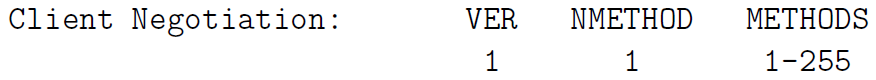
specify a target, the next rule will be considered.

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

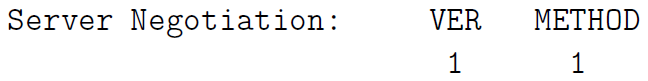
* Transport Layer: packet filtering. Examine every packet, check its IP headers and higher-level protocol headers to decide whether or not to let the packet through and whether or not to change any of the header fields.
* Application Layer: HTTP Proxy. examines the requested session for whether they 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: a generic proxy protocol for TCP/IP based network applications, consists of two components: A SOCKS client and a SOCKS server.

* socks client is implemented between 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, 1080.
* 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. Any response received back from the server is forwarded back to the LAN client.
* 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 its own IP address. So the servers on the internet side cannot see the actual IP addresses of the LAN hosts. With a SOCKS proxy is that your IP address would remain hidden even when it is a public address. Routing all network traffic through a proxy server makes it easy to centrally log all internet bound traffic and the caching of web services.
* 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, can access a remote DNS server. (with SOCKS4, the clients are required to resolve directly the IP addresses of the remote hosts, 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 socks server on port 1080.
     + client sends a Client Negotiation packet suggesting a set of different authentication methods that the server could use.



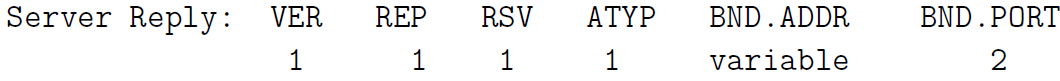
* + - * VER: the version number (SOCKS4 or SOCKS5)
      * NMETHOD: the number of methods
      * a listing of those methods by their ID numbers
    - server accepts the client packet and responds back with a Server Negotiation packet:



* + - * METHOD field: authentication method that server wishes to use.
      * socks server 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 possible values:
      * 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. proxy server evaluates request, includes 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) into a socks client.
  + By setting the environment variable LD\_PRELOAD, socksify saves you from the trouble of recompiling your client application 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

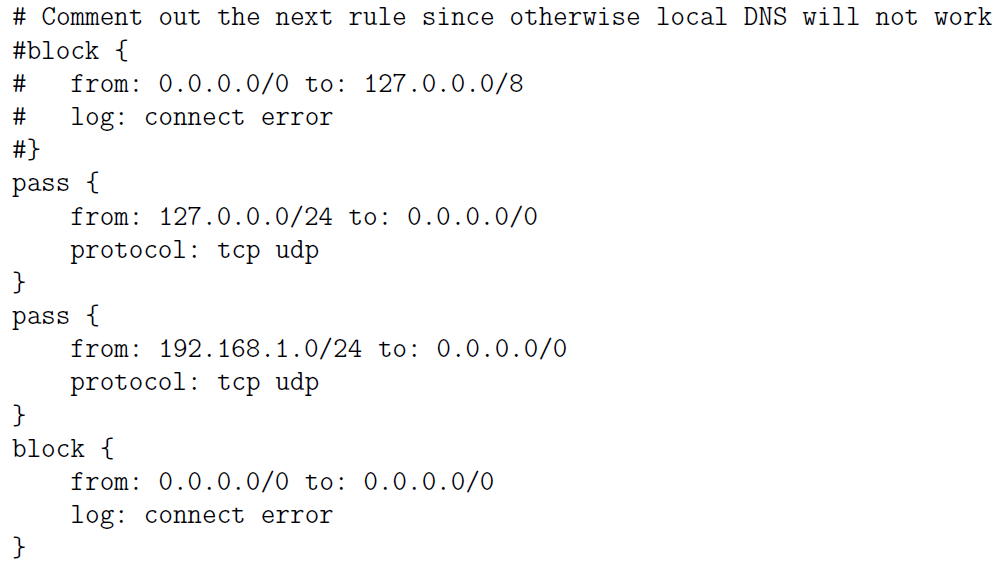
use a socks proxy behind a home or a small-business router wont gain any anonymity from the outgoing IP address used by the SOCKS server since the router will translate the outgoing (the source) IP address into what is assigned to router by the ISP anyway.

* + - * + 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.



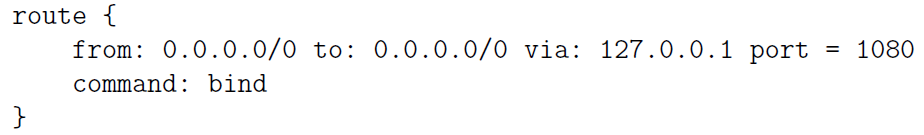
These rules say to allow all local socks clients on the same machine and all socks clients on the local LAN to talk to the SOCK proxy server on this machine. The third rule says to deny access to all other socks clients.

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

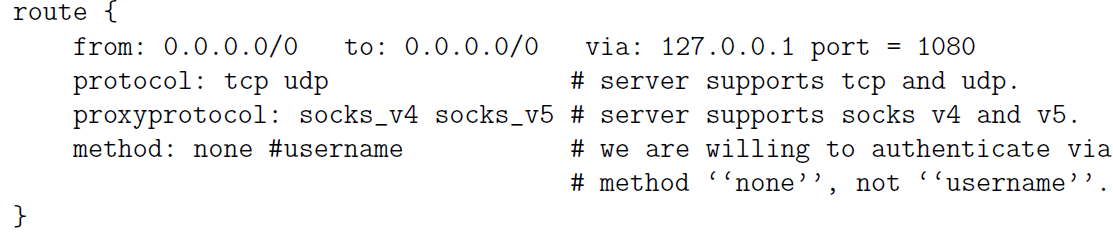


any local socks client will be able to call on any service anywhere for a TCP or UDP service. The third rule does the same for any socks client in the local LAN. The fourth rule blocks all other socks client requested services.

* + - * 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: if 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 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

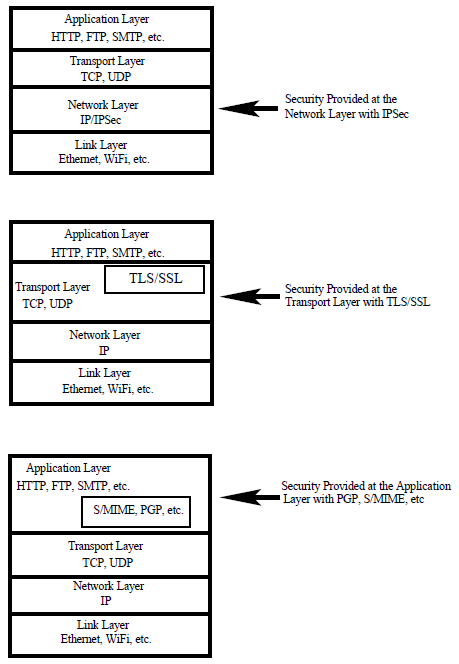
point to the receiving point in a network and sender is actually the entity that the information was received from.

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

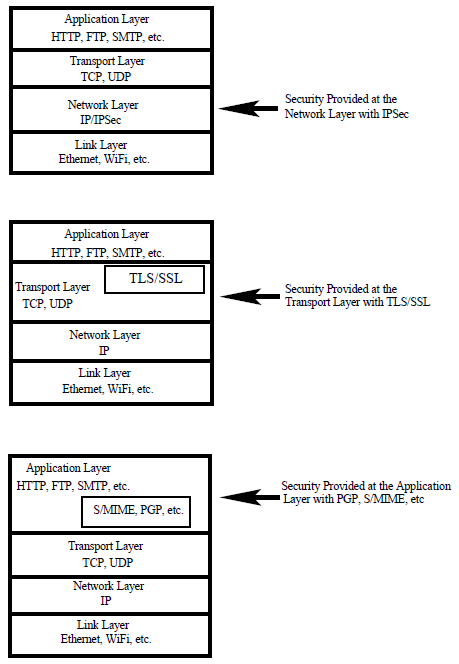
* Authentication:
  + the source is indeed as alleged in the information.
  + data integrity: The information was not altered along the way.
* Confidentiality: information is safe from being eavesdropped on during its transit from the sending point to the receiving point.
* Choosing the best security 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】

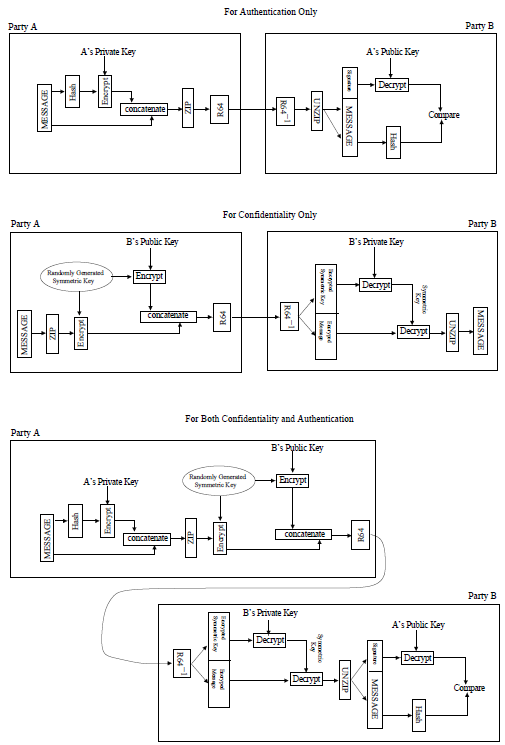
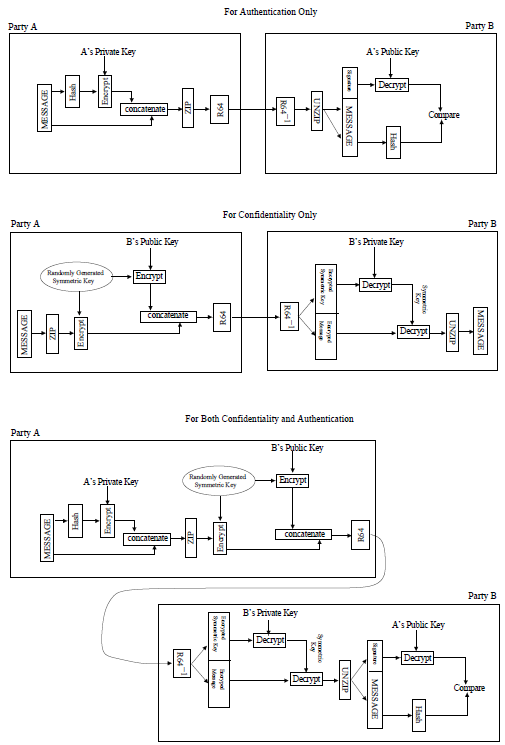


* In each of the three different layers mentioned above
  + authentication can be provided by public-key cryptography and key management issues in all layers can be made complicated by the fact that, in general, users are allowed to have multiple public keys.
  + confidentiality can be provided by symmetric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

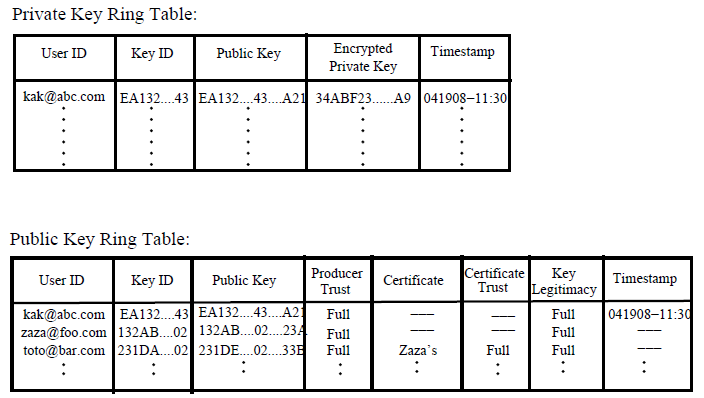
▓internet standards often use octet for a byte and not infrequently datagram for a packet.

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

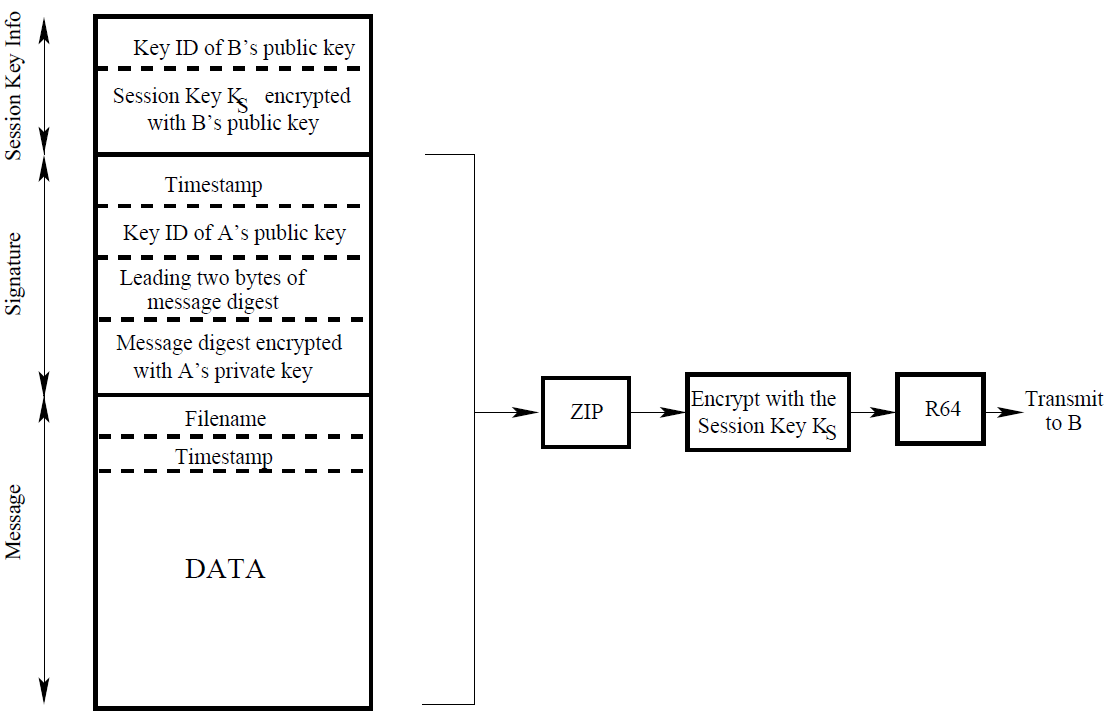
* PGPs operation consists of five services:
  + Authentication Service: sender attaching his/her digital signature to the email and the receiver verifying the signature using public-key cryptography.
    - RSA/SHA based digital signature
      1. At the senders end, a SHA hash function is used to create 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
  1. 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.
  2. 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.
  3. 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

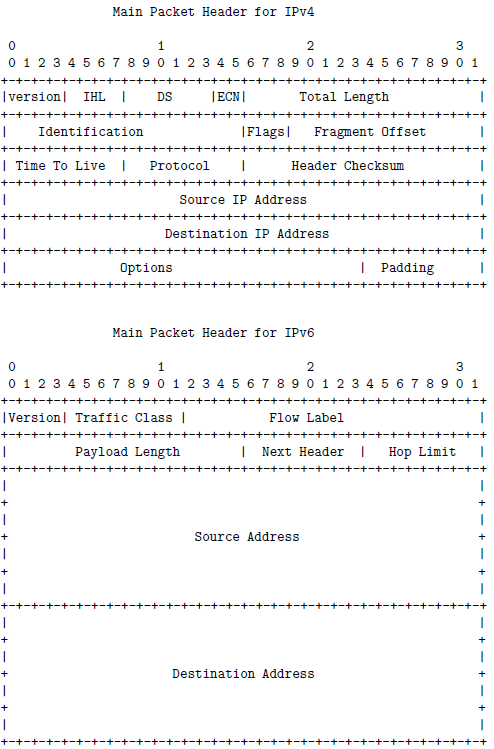


* PGP’s Web of Trust: A unique feature of PGP is its own notion of a certificate authority for authenticating the binding between a public key and its owner. Web of trust is a bottom-up approach to establishing trust for authentication.
  1. users public key can be signed by any other user, everyone 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.
  2. 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

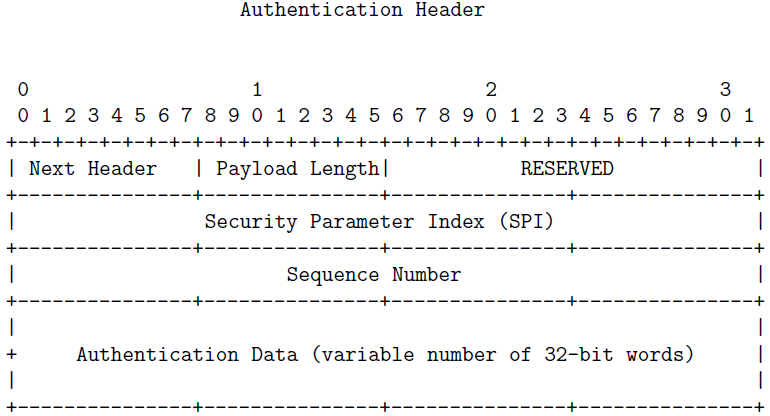
gpg --keyserver pgp.mit.edu --send-keys your\_8\_char\_KeyID

▓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:
  1. 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.
  2. 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.
  1. 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.
  2. 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.)



* 1. 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

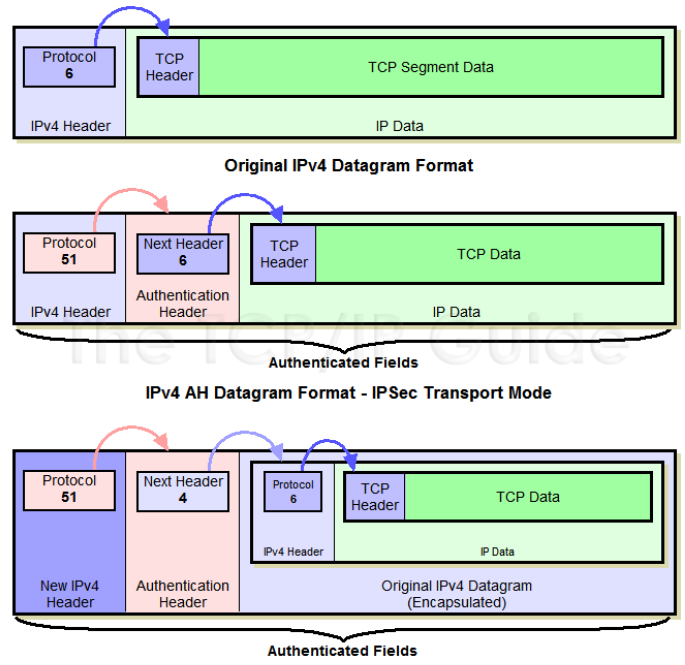


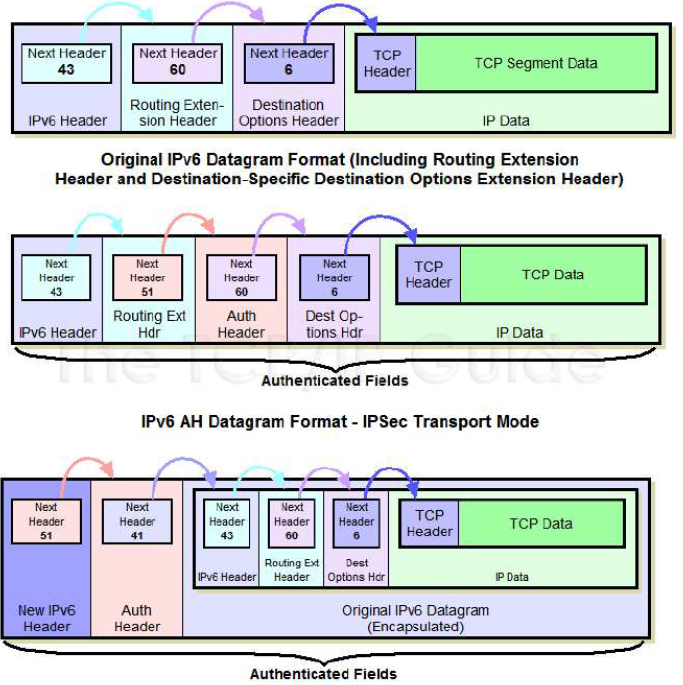
* + - IPv6 packet without AH header



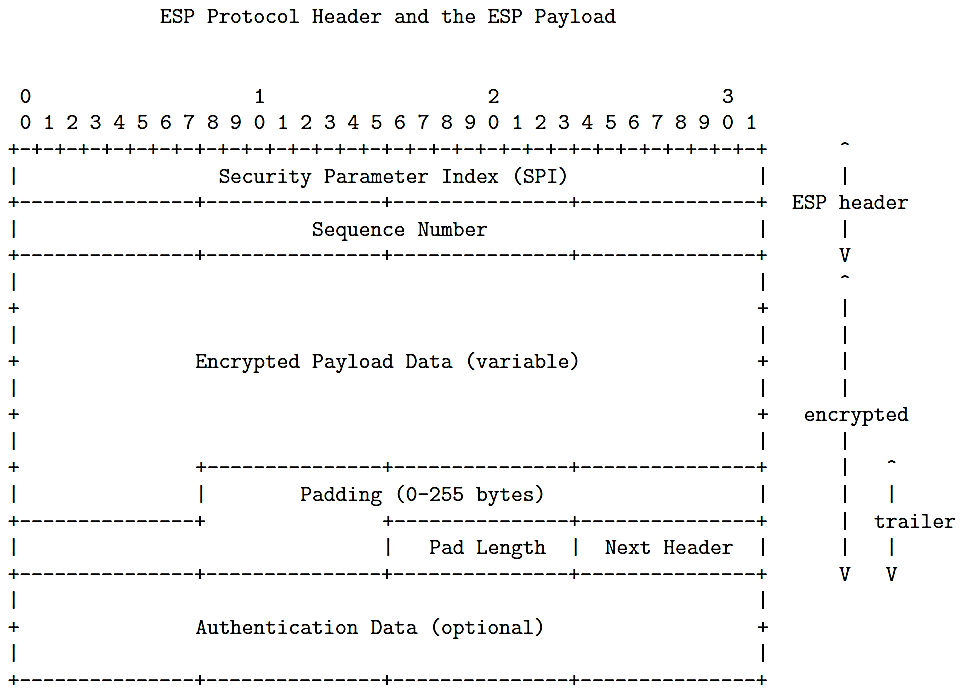
* + - IPv6 packet with AH header

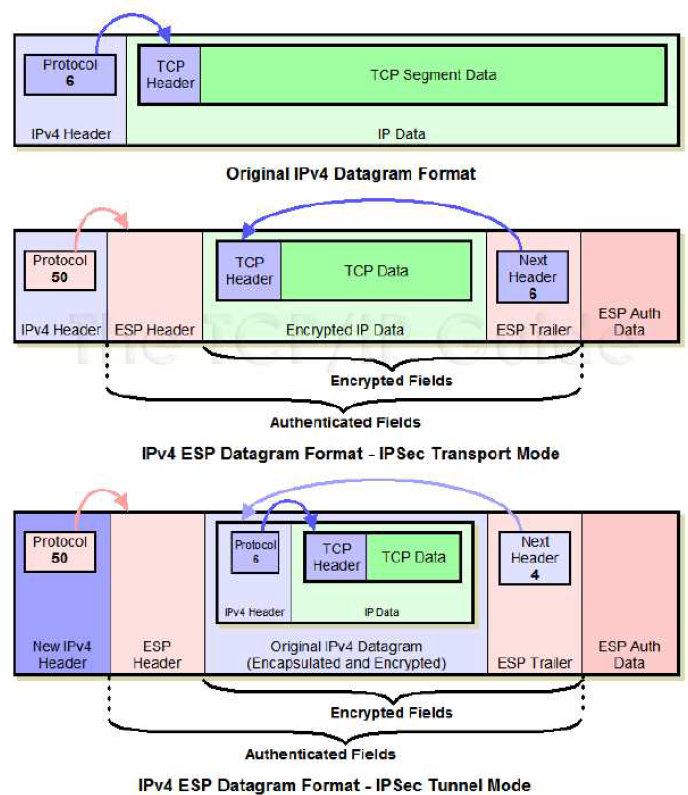


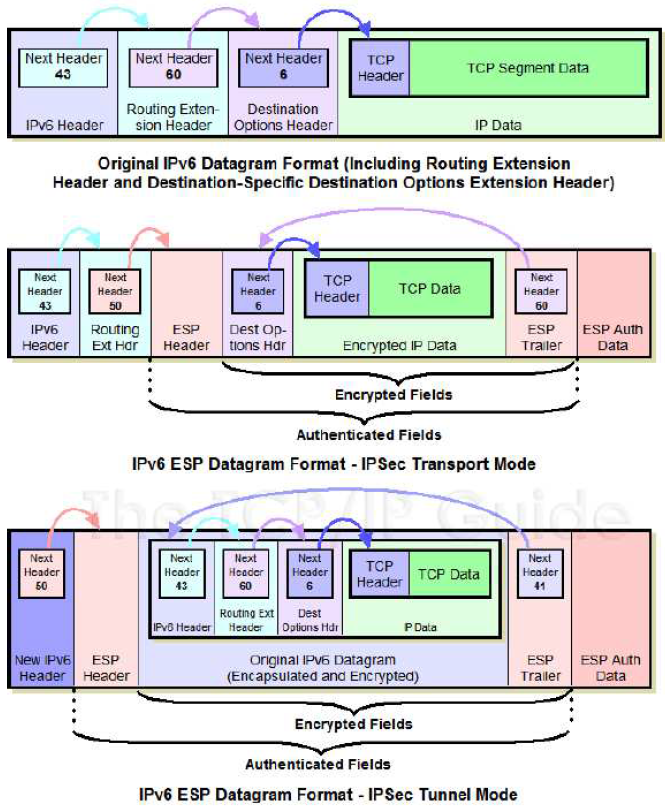




* 1. Encapsulating Security Payload (ESP) and Its Header: its main purpose is to provide confidentiality through encryption.
     + acronym ESP.:
       1. header: the first eight bytes.
       2. Payload: payload swallows up the TCP segment in the original IP packet. The encrypted version of the TCP segment (TCP header plus the data payload of the TCP segment)) is in the Encrypted Payload Data portion of the ESP payload.
     + the value of the Next Header field would contain number 6 and point backwards to the main content of the Encrypted Payload Data
     + adversary would not be able see even the field since it is a part of Next Header what stays encrypted in an ESP packet.
     + Padding: length of the encrypted segment would be a multiple of the block size used for encryption with symmetric key cryptography.
     + Pad Length: how much padding is used
     + Before encryption, an ESP Trailer is appended to the data to be encrypted. The payload (TCP/UDP message in the transport mode or the encapsulated IP datagram in the tunnel mode) and the ESP Trailer are both encrypted, but the eight-byte ESP Header is not.
     + In Transport Mode, ESP achieves confidentiality by placing in its an encrypted version of entire Encrypted Payload TCP segment, in Tunnel Mode, the payload contains an encryption of the entire IP packet and Encrypted Payload is obtained by encrypting the entire IP packet along with the padding and the ESP trailer as before.
     + Authentication Data field attached at the very end consists of the MAC value of the ESP packet. This value is known as Integrity Check Value
     + If the optional ESP authentication is used, the authenticator is calculated over the entire ESP datagram. This includes the ESP Header, the payload, and the trailer.







▓IPSec Key Exchange: the two endpoints first establish an SA (Security Association) that declares what authentication and encryption algorithms will be used and keys (ESP needs secret key, AH needs an authentication key) are exchanged with the Internet Key Exchange (IKE) protocol

* IKE combines the functions of three other protocols:
  + Internet Security Association and Key Management Protocol (ISAKMP): for exchanging encryption keys and security association information.
  + Oakley Key-Exchange Protocol: based on Diffie-Hellman algorithm but provides additional security, used by ISAKMP for creating a packet content encryption key.
    - Diffie-Hellmans computationally expensive modular exponentiation makes it vulnerable to a clogging attack in which an adversary may forge the source address of a legitimate party and send a public Diffie-Hellman key to an unsuspecting host, which then has to carry out modular exponentiation to compute the secret session key. But repeated receipts of the same request could clog up the host by causing it to spend all its time in modular exponentiation.
    - clogging attack is solved by using a cookie-exchange between the two parties. A request for a secret session key must be accompanied with a Cookie
    - Cookie exchange : each side sending a pseudorandom number to the other that must be acknowledged by the receiving party to the sending party. If the original requester for a secret session key was masquerading as someone else, they would never receive the cookie.)
    - Cookie: generated by hashing IP source and destination addresses, UDP source and destination ports, and a locally generated secret value.
    - Diffie-Hellman is also vulnerable to man-in-the-middle attack
  + SKEME protocol: for key exchange, ISAKMP uses the re-keying feature of this protocol.
* application of IPSec is in VPN:
  1. a typical VPN implementation provides you with a secure point-to-point tunnel between two specific endpoints in the VPN overlay network. Each SA is for just one communication link.
  2. Group VPN: the same SA is shared by a large collection of communication endpoints.

▓SSL/TLS FOR TRANSPORT LAYER SECURITY: E SSL/TLS plays a central role in the security and privacy needed for web commerce, your laptop must make certain that the remote host is indeed what it claims to be.

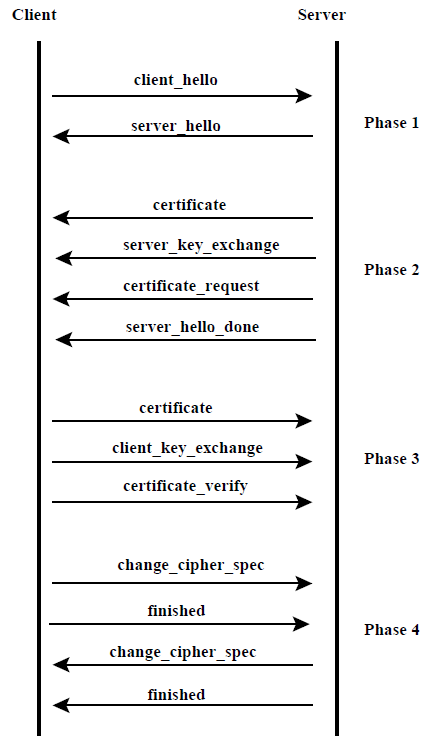
* Fundamental to the security that is established with the SSL/TLS protocol are the certificates issued by the Certificate Authorities (CA).
  1. server-only authentication:
     + client receives servers certificate, verifies servers certificate, generates a secret key that it then encrypts with the servers public key.
     + client sends the encrypted secret key to the server; the server decrypts it with its own private key and subsequently uses the client-generated secret key to encrypt the messages meant for the client.
  2. server-client authentication: in addition to secret key, the client also sends to the server its certificate that the server uses for authenticating the client
* OpenSSL is an implementation of the SSL and the TLS protocols. OpenSSL is used by the HTTPS and SMTPS protocols.
* SSL is composed of four protocols in two layers
  1. SSL Handshake Protocol: authenticates clients and servers to each other
     + Generate information about algorithms to use for compression, authentication, and encryption in SSL Record Protocol.
     + Generate cryptographic keys to be used for the encryption and the authentication of each SSL record.
     + SSL Handshake protocol works in four phases.
       1. Handshaking: initiated by the client, is used to establish the security capabilities present at the two ends of a connection.

The client sends to the server a client hello message with the following parameters:

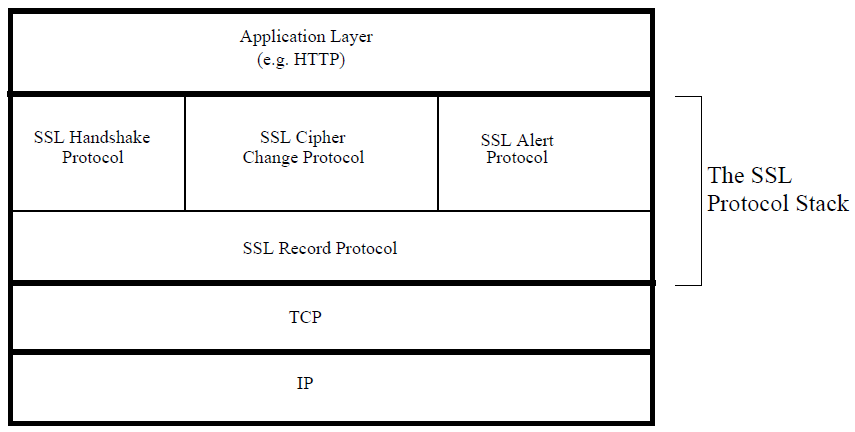
* + - * 1. Version (the highest SSL version understood by the client)
        2. Random (a 32-bit timestamp and a 28-byte random field that together serve as nonces during key exchange to prevent replay attacks)
        3. Session ID (a variable length session identifier)
        4. Cipher Suite (a list of cryptographic algorithms supported by the client, in decreasing order of preference);
        5. Compression Method (a list of compression methods the client supports)

server responds with its server hello message that has a similar set of parameters. Cipher Suite parameter in the server hello message consists of two elements.

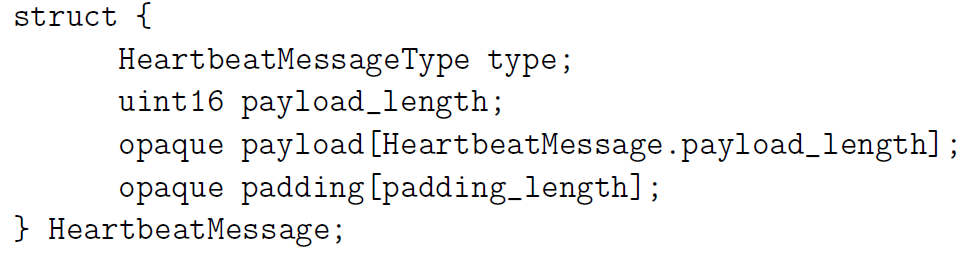
1. Key exchange method: RSA, Diffie-Hellman, etc.
2. CipherSpec: indicate the authentication algorithm selected, the length of MAC, the encryption algorithm, etc.
   * + 1. initiated by the server by sending the client a message labeled certificate containing its one or more certificates for the validation of the server public key. This could be followed by a server key exchange message, and a certificate request message if the server also wants to validate the client. Phase 2 handshaking ends when the server sends the client a server hello done message.
       2. initiating by client by sending to server the clients certificate (message labeled certificate) only if server made a request). Next, the client sends to the server a mandatory client key exchange message that could consist of a secret session key encrypted with the servers public key. This phase ends when the client sends to the server a certificate verify message to provide a verification of its certificates if they are signed by a certificate authority.
       3. completes the setting up of a secure connection between the client and the server. The client sends to the server a change cipher spec message indicating that it is copying the pending CipherSpec into the current CipherSpec. Next, the client sends to the server the finished message and the server does the same vis-a-vis the client.



* 1. SSL Record Protocol: transmits the data confidentially. Sits directly above the TCP protocol, provides Confidentiality and Message Integrity. Fragment the data into blocks, applying authentication and encryption primitives to each block (SSL record: Each output block produced by the SSL Record Protocol), and handing the block to TCP for transmission over network. SSL Record Protocol consists of the following five steps:
     + Fragmentation: message is fragmented into blocks whose length does not exceed 2^14 bytes.
     + Compression: optional step requires lossless compression and carries the stipulation that the size of the input block will not increase by more than 1024 bytes.
     + Adding MAC: computes the MAC and appended it to the compressed message block.
     + Encryption: compressed message and the MAC are encrypted using symmetric-key encryption.
     + Append SSL Record Header: an SSL header is prepended to encrypted block. The header consists of 8 bits for declaring the content type, 8 bits for declaring the major version used for SSL, 8 bits for declaring the minor version used, and 16 bits for declaring the length of the compressed plaintext (or the plaintext if no compression was used).
  2. SSL Cipher Change Protocol: change cipher spec message format must correspond to the Change Cipher Spec Protocol. Message must consist of a single byte with a value of 1 indicating the change.
  3. SSL Alert Protocol: used to convey SSL-related alerts to the peer entity.



* a connection takes care of transferring information securely from one endpoint to the other, the concept of a session allows for such data transfers to take place back and forth without having to renegotiate the security parameters for each separate connection.
* SSL Connection: connection is a one-time transport of information between two nodes in a communication network.
  1. A connection constitutes a peer-to-peer relationship between two nodes.
  2. Being one-time, connections are transient.
  3. Every connection is associated with a session.
  4. SSL connection state is characterized by the following parameters:
     + Server Write MAC Secret: used in calculating the MAC (Message Authentication Code) value for the data sent by the server.
     + Client Write MAC Secret: used in calculating the MAC value for the data sent by the client.
     + Server Write Key: symmetric-key encryption key for data encrypted by the server and decrypted by the client.
     + Client Write Key: symmetric-key encryption key for data encrypted by the client and decrypted by the server.
     + Initialization vectors: for each key used by a block cipher operating in the CBC mode is maintained, initialized by the SSL Handshake Protocol. The final ciphertext block from each record is preserved for use as the IV with the following record.
     + Sequence Numbers: Each party maintains separate sequence numbers for the transmitted and received messages through each connection. When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero.
* SSL Session: session is an enduring association between a client and a server.
  1. A session is created by the SSL Handshaking Protocol.
  2. A session can consist of multiple connections.
  3. A session is characterized by a set of security parameters that apply to all the connections in the session.
  4. A session comes to an end when the exchange of data between the two endpoints has come to an end.
  5. SSL session state is characterized by the following parameters:
     + Session Identifier: An arbitrary byte sequence chosen by the server to identify an active or resumable session state.
     + Peer Certificate: X509.v3 certificate of the peer.
     + Compression Method: algorithm to compress data prior to encryption.
     + Cipher Spec: bulk data encryption algorithm and the hash algorithm used for MAC (Message Authentication Code) calculations
     + Master Secret: 48-byte secret shared between the client and server.
     + IsResumable: flag indicating whether the session is allowed to initiate new connections.
* Heartbeat Extension to the SSL/TLS protocol: sits on top of the SSL/TLS Record Protocol, leave a session open in anticipation of upcoming data exchanges between the two endpoints
* Since there is significant overhead associated with the negotiation of the security parameters for establishing a secure session, some applications may require that once the security parameters have been agreed upon through the SSL/TLS Handshake protocol, they should continue to hold good even through lulls in data exchange between the two endpoints.
* Central to the Heartbeat Extension Protocol are two messages: one endpoint sends a HeartbeatRequest message to the other endpoint, the other send back a HeartbeatResponse message. A Heartbeat Request message may arrive at any time during the lifetime of a session.
* When one endpoint sends a HeartbeatRequest message to the other endpoints, the former also starts retransmit timer. During time interval of retransmit timer, sending endpoint won’t send another HeartbeatRequest message. An SSL/TLS session is considered to have terminated in the absence of a HeartbeatResponse packet within a time interval.
* Heartbeat Extension protocol also includes Heartbeat Hello Extension that an endpoint can use to inform the other endpoint whether its implementation supports Heartbeats. An endpoint can also indicate whether it is only willing to send HeartbeatRequest messages, or only willing to accept HeartbeatResponse messages, or both.
* As a protection against a replay attack, a HeartbeatRequest packet must include a payload that must be returned without change by the receiver in its HeartbeatResponse packet. Heartbeat protocol specifies that a request packet include values for the following two fields: an arbitrary payload and an integer that specifies the length of the payload. The protocol also specifies that the payload must be followed by padding (an arbitrary sequence of bytes) whose length must be at least 16 bytes. The padding bytes are ignored by the receiving endpoint.
* protocol specification for a Heartbeat message: first field, of size one byte, specifies whether it is a HeartbeatRequest message or a HeartbeatResponse message.



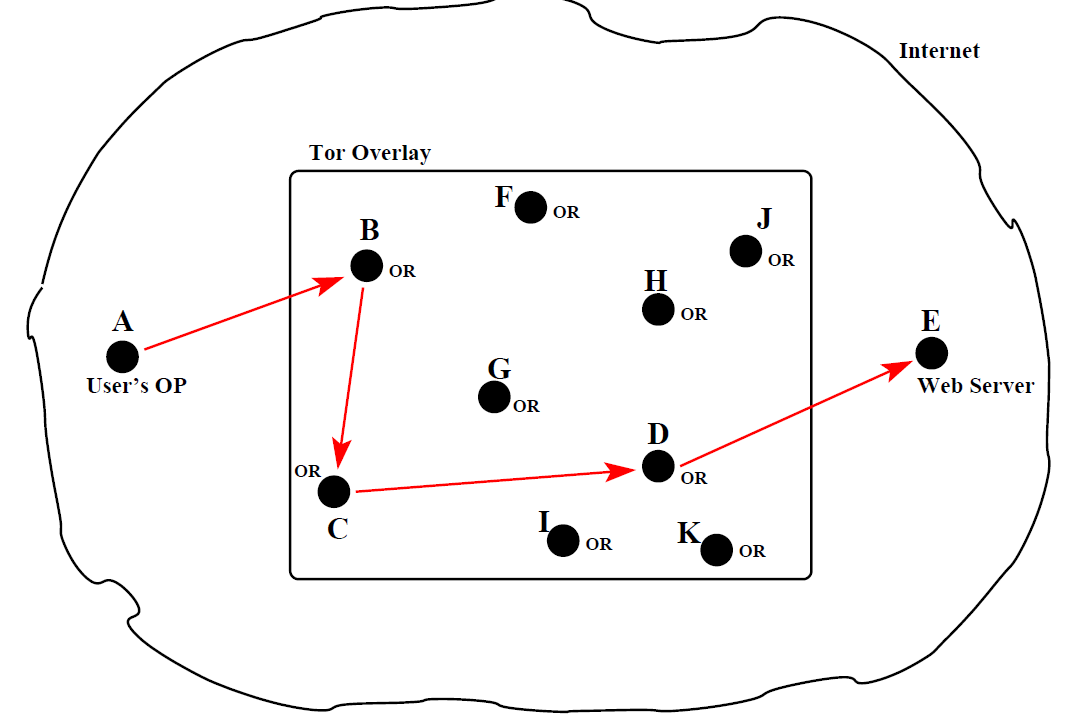
▓traffic analysis attack: Gleaning information regarding the original source

of the packets and their ultimate destination. The packet headers are

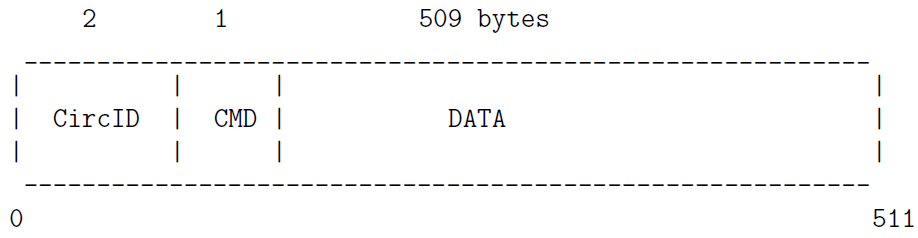
always in clear text.

▓Tor (The Onion Router) protocol for anonymized routing: to figure out a way to set up internet communications so that an adversary would not be able to analyze packet headers for the purpose of finding out who was talking to whom.

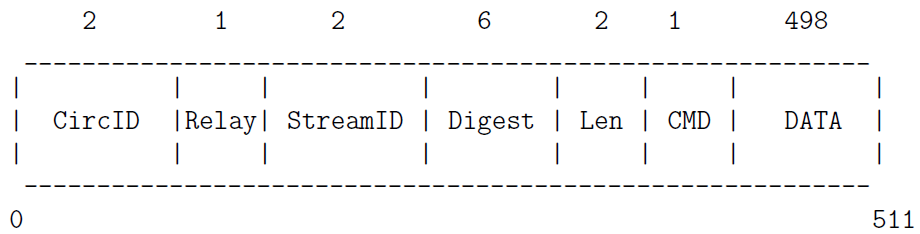
* What makes the Tor protocol work is a very clever interplay between the RSA public-key cryptography and the DH (Diffie-Hellman) public-key cryptography.
* Tor protocol is based on the twin notions of Onion Proxies (OP) and Onion Routers (OR). A users OP first queries a Tor directory for the IP addresses of the ORs in the Tor overlay, then selects a subset of these ORs, commonly just 3, for constructing a path to the destination resource.
* the routing knowledge at any single node on a path through the Tor overlay is limited to exactly two nodes, the immediately preceding node on the path and the immediately following node
* A users OP constructs a path through the Tor overlay. This path constitutes a circuit. Subsequently, the two parties at the two end of a circuit may use it for an arbitrary number of TCP streams.
* Example: B, C, D are the ORs selected by user A for a path to the destination E.



* a message that is exchanged between an OP and an OR or between two ORs is called a cell / torpacket. Each torpacket consists of 512 bytes and the control and the relay torpackets work together to create an end-to-end path (meaning a circuit) in the Tor overlay in such a way that each interior node on the path has only local knowledge of the path.:
  + control torpackets: alter the relationship between the sender node and the next node on the path that receives such a packet.



* can be of the following kinds: create, created, destroy, and padding.
* circID : a circuit identifier is unique to each hop in a circuit despite the fact that the circuit abstraction applies to entire end-to-end path.
* CMD: a one-byte integer representation of a command. A control torpacket may contain the following different commands:
  + create : sent by an OP or OR to another OR to extend the path to the next node
  + created : when an OR successfully extends the path to the next node in response to a create command from the previous node on a path, it sends back a created message to the previous node.
  + destroy : sent by a node to another node to teardown the path
  + padding : used for keepalive when a timeout might shut down a circuit otherwise
  + relay torpackets: carry the data that is exchanged between the two endpoints, that can only be done after a path is fully constructed. During the process of path construction, the data carried by relay torpackets is for the purpose of extending the path beyond the current termination point. Such relay torpackets generate control torpackets at the current terminal node on the path for extending the path. last node on the path.



* byte command field (CMD) in the header of a relay torpacket can be used to create following kinds of such packets:
  + relay extend : to extend the circuit by one hop
  + relay extended : to notify that relay extend was successful
  + relay truncate : to drop the last the OR on the path
  + relay truncated : to notify that relay truncate was successful
  + relay begin : to open a new stream
  + relay connected : to notify OP that a stream is opened
  + relay end : to close a previously opened stream
  + relay data : for transmission of data in stream
  + relay sendme : used for congestion control
  + relay teardown : used to close a broken stream
  + how a users OP uses the control torpackets to create an end-to-end circuit incrementally
    - every OR node has a public RSA key that it makes available to the users OP. These public keys will be static. So any communication sent to an OR that is encrypted with its RSA public key can only be understood by that OR.
    - DH Y keys (Diffie-Hellman public key) are created on the fly between the users OP and each of the ORs on the path chosen by the user. The purpose of the DH Y keys is that when the users OP wants to send a message to a designated OR on the path, it is encrypted with the session key derived from the OPs DH Y key and that ORs DH Y key.
    - Example: source A ->B ->C ->D -> destination E
      1. The users OP sends a create control torpacket to the first node in the path chosen by the user. A’s OP sets the CircID field of this torpacket to a new value, circID\_AB, that was not previously used. The DATA field of this packet contains A’s DH Y key Y\_AB encrypted with B’s RSA public key.
      2. B responds back to A with the created control torpacket. The DATA field of this torpacket contains Bs DH Y key Y\_BA. Now both A and B can calculate the secret session key K\_AB for their link
      3. Nodes A and B can now start exchanging relay torpackets, all using the identifier circID\_AB for the circID field. In order to extend the circult, A sends B a relay torpacket with the relay extend command. The DATA field of this relay extend torpacket includes a DH Y key Y\_AC that is meant for the new terminal node C on the path. In order to make sure that the key Y\_AC is not seen by node B, it is encrypted with C’s RSA public key. The DATA field in the relay extend torpacket from A to B is encrypted with the session key KAB.
      4. When B receives the relay extend torpacket from A, it knows that it is the current endpoint on the path. So it generates a control torpacket whose DATA field contains A’s DH Y key Y\_AC encrypted with C’s RSA public key. The control torpacket sent by B to C uses a new randomly generated number for the circID field, circID\_BC. This becomes the identifier for the segment of the circuit between the nodes B and C. Only the node B knows both circIDAB and circIDBC. This fact plays an important role in ensuring that each node on the path has only the local knowledge of the path.
      5. Node C responds back to B with a created control torpacket. The DATA field of this torpacket contains C’s DH Y key Y\_CA meant for A. Node B sends this acknowledgment back to A using the relay extended torpacket, with its DATA field containing the key Y\_CA. Now both A and C can calculate the secret session key K\_AC for any messages that A may want to send to C (through B) that B is not allowed to see.
      6. The path may be extended in the same manner to the node D
      7. In constructing an end-to-end circuit in the manner described above, there was never a need for using As public RSA key. User A remains anonymous to all the ORs in the circuit. B will remain anonymous to D and so on. But all the ORs in a circuit are known to the user A
      8. A sends a relay begin torpacket to B, from where it is forwarded to the next node on the circuit, and so on, thus creating an end-to-end stream between A and E. The user A is allowed to create an arbitrary number of streams and they can all share the same circuit. While the different TCP streams will have different streamID values in the relay torpackets that carry the stream data, they will have the same value for the circID field
      9. After an end-to-end circuit is created in this manner, the user A can start pushing data into the circuit that is meant for the final destination E.
      10. the stream data that the user A places on the wire is encrypted with the K\_AD session key, followed by its encryption with K\_AC session key, followed by its encryption by K\_AB session key. [Hence onion.] As these stream data bearing relay data torpackets are received by B from A, the node B uses the session key KAB to decrypt the top layer of encryption and forward the stream to the next code, node C, in the circuit. This process continues until the stream data reaches the final node D, from where it goes via the normal TCP transmission to the application running at the destination E.
  + Can the exit node operator see the source IP address (IP address of node A)? It should not be possible.Tor logic that keeps As IP address shielded from the exit node D is the same as the logic that keeps Bs IP address shielded from D. The packets that go out from D to the web server at E should only bear Ds IP address in the source fields. When D receives replies to those packets from the web server, it simply forwards them back to C.
  + Can the exit node operator see the data payload of the source packet? If node A is trying to reach an HTTPS web site, that implies end-to-end encryption of the payload in the packets. In that case, the exit node operator obviously cannot peer inside the packets that A is sending out.

▓daemon / daemon: A continuously running computer program that provides a service to others in a network

▓buffer underflow vulnerability: occurs when the temporary holding space during data transfer, the buffer, is fed at a lower rate than it is being read from. when the flow of data from the original source, typically the hard drive, was interrupted long enough for the buffer to reach full capacity and empty itself. As a result, the writing action is stopped and the device receiving the data may be ruined

▓Telnet protocol: allows user to establish a remote terminal sessions created with SSH command on a remote machine for the purpose of executing commands there.

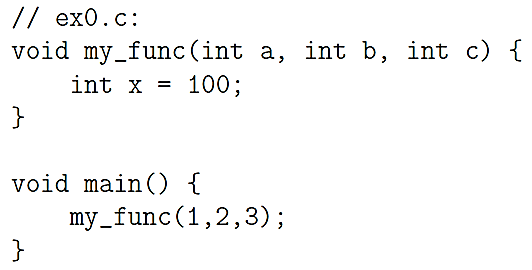
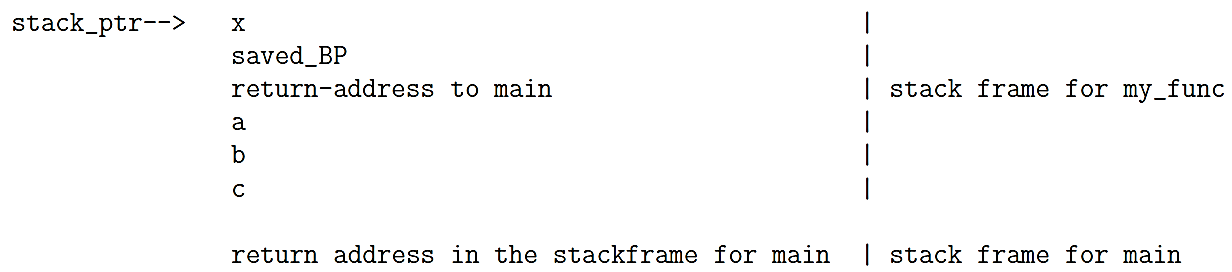
* a constantly running telnetd daemon at a Telnet server monitors port 23 for incoming connection requests from Telnet clients.
* When a client seeks a Telnet connection with a remote server, the client runs a program called telnet that sends to the server machine a socket number, which is a combination of the IP address of the client machine together with the port number that the client will use for communicating with the server.
* server receives the client socket number, it acknowledges the request by sending back to the client its own socket number.

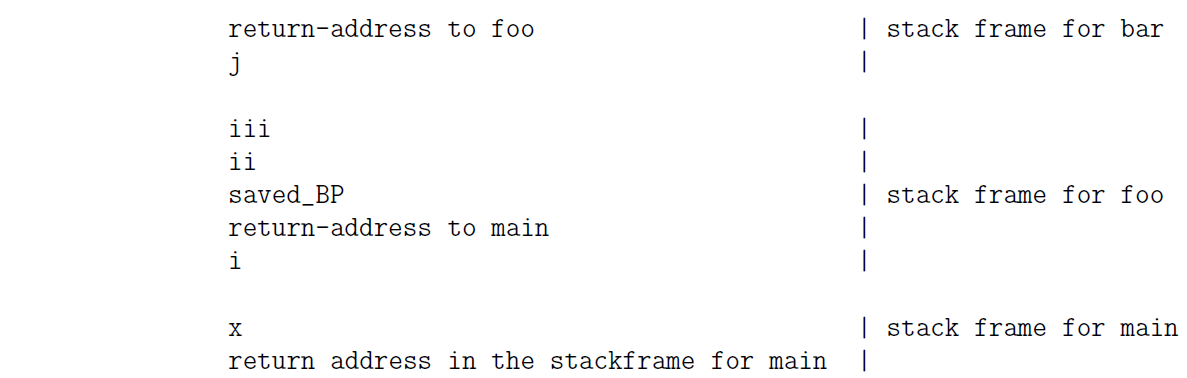
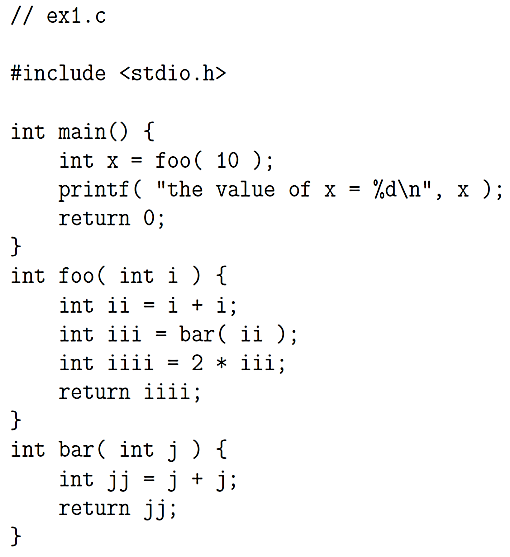
▓buffer overflow: occurs on the stack when information is written into the memory allocated to a variable on a stack but the size of this information exceeds what was

allocated at compile time.

* Since return addresses to are not stored in heaps, it is more difficult to launch functions exploits with heap overflows than with stack overflows. Therefore, a stack buffer overflow is far more likely to be the cause of a security vulnerability than a heap overflow
* a stack overflow can be used to overwrite location where “return address to a function” is stored and that can send execution into a piece of malicious code.

▓When you run an executable, it is run in a process. Every process is assigned a stack. As it encounters each new local variable, it is pushed into the stack, and as it encounters a function call, it creates a new stackframe on the stack.

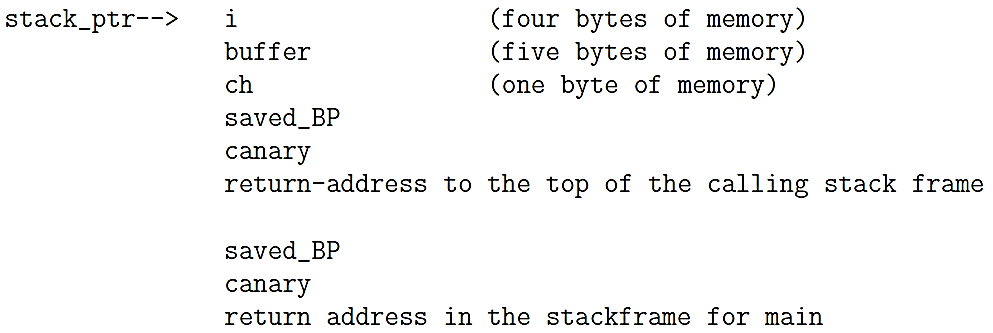
 

These stack actions call for the third argument to be pushed into the stack, followed by the pushing of the second argument, and then the first argument. Subsequently, there is the call to my\_func. This last action pushes the current content of the Instruction Pointer (IP) into the stack, where it becomes the return address for the calling function in the stack frame for my\_func. The call to my\_func also causes the current content of the Base Pointer to be pushed into the stack (saved\_BP). when the current stackframe finishes execution, we must quickly restore the Base Pointer to the value for the calling stackframe.

the call stack consists of a sequence of stackframes, one for each calling function that has not yet finished execution, topped by the stackframe for the function currently undergoing execution. The return address you see in each stackframe is the memory address of the calling function. The values stored in each stack frame above the location of the return address are for those local variables that are still in scope

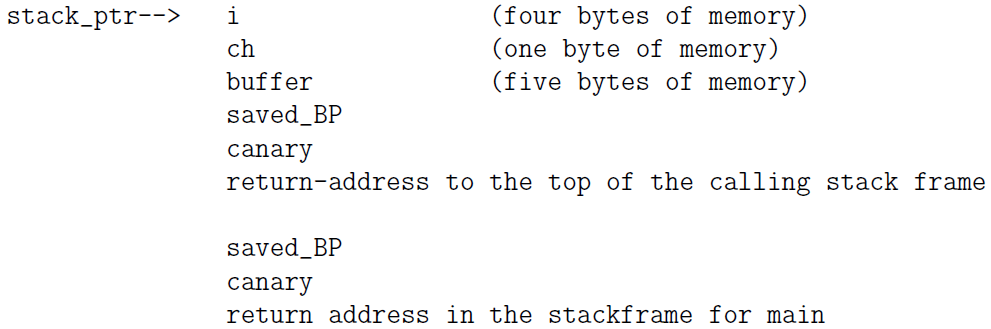
at the current moment.

* Stack Pointer: register always points to the top of the process call stack.
* Base Pointer / Frame Pointer: points to the base of the current stackframe, this address stays fixed as long as the flow of execution is in the current stackframe. This allows for efficient memory dereferencing for accessing the function call parameters and the local variables in the function corresponding to the current stack frame. Note that these parameters and variables remain at fixed distances vis-a-vis the memory address pointed to by the Base Pointer regardless of push and pop operations on the stack.
* Instruction Pointer: holds address of the next CPU instruction to be executed.
* by convention the stack grows downwards, as the stack grows, the addresses go from high to low. So when you push a 4-byte variable into the stack, the address to which the stack pointer will point is previous value minus 4.
* Buffer Overflow Attack: Overrunning the Memory Allocated on the Call Stack. EX. the string you enter will begin to overwrite the memory locations allocated to other variables on the stack and also possibly the location where the return address of the calling function is stored. When this happens, the program will be aborted with a segmentation fault.
* The basic idea that is used in several buffer overflow protection algorithms is a combination of rearrangement of the local variables on the stack and the insertion of a special variable, canary, just below the stack locations reserved for the local variables.
* For any given function
  + code segment generated by prologue: reserve memory for the local variables on the call stack
  + code generated by the epilogue: clean up stack frame just before function is done and the flow of execution has returned to the calling function.
* When stack protection is needed, the code generated by the prologue also inserts a special location in the stackframe where a guard value is stored. This location in a stackframe is called a canary and any change in the guard value stored there taken as an attempt at buffer overflow exploitation.



An attacker would not want to change the value of the canary since the epilogue would detect that immediately and cause the process to abort. Attacker would have to create an overflow string that incorporates the sequence of characters 0x000aff0d. But now the C library function strcpy() and gets() for changing the return address would not work. That is because strcpy() will not be able to get past the null byte in the attackers overflow string and gets() wont be able to get past the newline character.

* Protection against buffer overflow exploits can be created by a function prologue that rearranges local variables in a stackframe so that the scalar variables are above the array variables in the stack



Now any overflow in the memory allocated to the buffer variable will not corrupt the scalar variables i and ch. Should there be any overflow in the value being stored in buffer, it will affect the canary. The saved frame pointer would

Saved\_BP still be vulnerable. However, by having the prologue code move the the canary to a location immediately above saved frame pointer saved\_BP, we could protect that also.

* buffer overflow vulnerability in some application software means
  + at least one function that requires a string input at run time
  + when this function is called with a specially formatted string, that would cause the flow of execution to be redirected in a way that was not intended by the creators of the application.

▓defense overflow attack

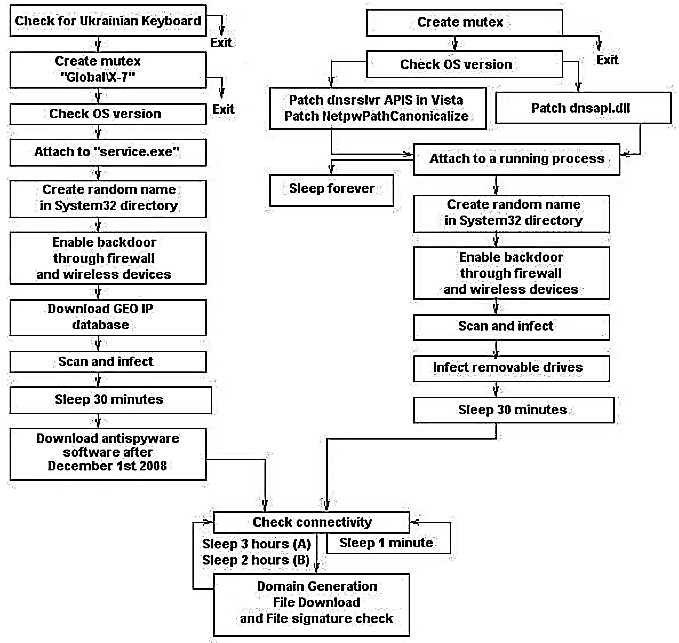
* If a buffer overflow attack calls for inserting the shellcode directly into the stack and executing it there, that can be thwarted by making the stack nonexecutable.
  + using the NX (No-eXecute) bit in a memory address:In 64-bit x86 processors, the bit at position index 63 (the most significant bit) serves as the NX bit. If this bit is set to 1, code starting at that position will not be executed by the processor.
* return-to-libc attack: the return address in a stackframe is replaced by the address of a library function that is already in the address space of the process
  + ASLR (Address Space Layout Randomization): with this module, its addresses are shifted by a random number. The process spawned for running the program will use a newly generated value for the random number each time, so it virtually impossible to associate a fixed process memory address with the standard functions

▓virus: malicious piece of executable code that propagates typically by self-replicating and attaching itself to a host document that will generally be an executable file.

* Typical hosts (a document or a file) for computer viruses
  + Boot sectors on disks and other storage media
    - When you turn on a computer, it starts executing the instructions starting at a designated memory address. These instructions tell the system what device to use for booting. Usually, this device is a disk that contains a specially designated region at its beginning that is called the boot sector.
    - boot sector: has the partition table for the disk and bootstrap code (boot loader) for pulling in the operating system at system boot time. The first sector in every disk partition serves as a boot sector for that partition; this boot sector is the Volume Boot Record (VBR).
    - Since boot sector code is executed automatically to enable the runtime memory allocator to figure out how to use those partitions for information storage, it is a common attack vector for viruses.
    - protection against boot sector corruption is to prevent System BIOS from writing to the first sector of a disk and the first sector of a disk partition.
    - boot sector viruses: Viruses that attach themselves to boot sectors
  + Executable files for system administration: file infector viruses (such as the batch files in Windows machines, shell script files in Unix, etc.) Documents that are allowed to contain macros: macro viruses. (such as PDF files, Microsoft Word documents, Excel spreadsheets, etc.). Macro programming capability can be exploited for creating executable code that acts like a virus. If a template that has been infected with malicious macros, all documents created from such a template will also be infected.
  + Any operating system that allows third-party programs to run
* a virus will duplicate itself when it attaches itself to another host document (executable file). In order to make more difficult its detection by pattern matching, a virus may alter itself when it propagates from host to host. Viruses that are capable of changing themselves are called mutating viruses.
* Viruses typically place a signature (such as a string that is an impossible date) at a specific location in the file to know if a potential host is already infected.
* To escape detection, the more sophisticated viruses encrypt themselves with keys that change with each infection. What stays constant in such viruses is the decryption routine.
* payload part of a virus is that portion of the code that is not related to propagation or concealment.

▓WORMS

* worm is self-contained, hop from machine to machine on its own and able to send copies of itself to other machines over a network. main difference between a virus and a worm is that a worm does not need a host document. Therefore, a worm can harm a network and consume network bandwidth, the damage caused by a virus is mostly local to a machine.
* A program may hop from one machine to another by means that include:
  + using the remote shell facilities provided by, say, ssh, rsh, etc., to execute a command on the remote machine and install a small bootstrap program on the target machine
  + cracking passwords and logging in as a regular user on a remote machine: extent of harm that a worm can carry out would depend on the privileges accorded to the guise under which the worm programs are executing.
  + using buffer overflow vulnerabilities in networking software: In networking with sockets, a client socket initiates a communication link with a server by sending a request to a server socket that is constantly listening for such requests. If the server socket code is vulnerable to buffer overflow or other stack, attacker could manipulate that into the execution of certain system functions on the server machine that would allow the attackers code to be downloaded into the server machine.
  + even when no local harm is done, a propagating worm can bog down a network and, if the propagation is fast enough, can cause a shutdown of the machines on the network. This can happen particularly when the worm is not smart enough to keep a machine from getting reinfected repeatedly and simultaneously. Machines can only support a certain maximum number of processes running simultaneously.
* Morris worm used the following three exploits to jump over to a new machine:
  + A bug in the popular sendmail program that is used as a mail transfer agent by computers in a network. when this worm attack, it sends a message to the sendmail program running on a remote machine with the name of an executable as the recipient of the message. The sendmail program, if running in the debug mode, would then try to execute the named file, the code for execution being the contents of the message. The code that was executed stripped off the headers of the email and used the rest to create a small bootstrap program in C that pulled in the rest of the worm code.
  + A bug in the finger daemon of that era.: A malicious program can exploit this feature to create fake stack frames and cause the rest of the program execution to be not as originally intended. If an executing program allocates memory for a buffer on the stack, but does not carry out a range check on the data to make sure that it will fit into the allocated space, you can easily encounter buffer overflow problem
  + use remote shell program rsh to enter other machines using passwords: break into a user account and harvest the addresses of the remote machines in their .rhosts files.
* SlammerWorm: In the UDP packet sent by the Slammer worm to a remote machine, the first byte 0x04 was followed a long string of bytes and did not terminate in the null character, so the information written into the stack would exceed the 128 bytes of memory reserved for the SQL server request. It is in the overwrite portion that the Slammer executed its network hopping code. It created an IP address randomly for the UDP request to be sent to another machine. This code was placed in a loop so that the infected machine would constantly send out UDP requests to remote machines selected at random.
* Conficker worm / Downadup / Kido.: the worm infection spread by exploiting a vulnerability in the executable svchost.exe on a Windows machine.
  + svchost.exe: svchost process replicates itself for each dynamically-linkable libraries DLL that needs to be executed. So any DLL that needs to be executed must attach itself to the svchost process.
  + generic host process: process executing the svchost.exe file
  + at system boot time, the svchost process checks the services part of the registry to construct a list of services (meaning a list of DLLs) it must load.
  + Remote Procedure Calls: With RPC, one machine can invoke a function in another machine without having to worry about the intervening transport mechanisms that carry the commands in one direction and the results in the other direction.
  + three different ways for the worm get to a computer
    - MS08-067 mode of propagation: A machine running a prepatched version of Windows Server Service svchost.exe could be infected because of a vulnerability with regard to how it handled remote code execution needed by the RPC requests coming in through port 445 which is assigned to the resource-sharing SMB protocol that is used by clients to access networked disk drives on other machines and other remote resources in a network. So if a machine allowed for remote code execution in a network, it would be open to infection through this mechanism. When a machine received a crafted string on its port 445, machine would
      * download a copy of the worm using the HTTP protocol from another previously infected machine and store it as a DLL file
      * execute a command to get a new instance of the svchost process to host the worm DLL
      * enter appropriate entries in the registry so that the worm DLL was executed when the machine was rebooted
      * gave a randomly constructed name to the worm file on the disk
      * continue the propagation:
        1. Windows API function NetpwPathCanonicalize() exported by netapi32.dll over an SMB session on TCP port 445 to canonicalize a string
        2. in an SMB session, this function was supplied with a crafted string by a remote host, it was possible to alter function’s return address in stack frame for function being executed. Attacker then used the redirected return address to invoke a function URLDownloadToFile() to pull in the worm file.
        3. Once the worm file had been pulled into the machine, it could be launched in a separate process/thread as a new instance of svchost.exe by calling the LoadLibrary() function whose sole argument was the name of the newly downloaded worm file. The LoadLibrary command also copied the worm file into the system root.
    - NetBIOS Share Propagation Mode: Once a machine was infected, the worm could drop a copy of itself in the hard disks on the other machines mapped in the previously infected machine. If it needed a password to drop a copy of itself at these other locations, the worm came equipped with a list of 240 commonly used passwords. If it succeeded, the worm created a new folder at the root of these other disks where it placed a copy of itself.
    - USB Propagation Mode: worm drop a copy of itself as the autorun.inf file in USB-based removable media. This allowed the worm copy to execute when the drive was accessed (if Autorun was enabled).
  + worm could update itself through its peer-to-peer communication abilities.
  + the worm was capable of resetting the system restore points, so an infected machine could be restored to good health by simply rolling back the software state to a previously stored system restore point
  + disassembler-inferred control-flow diagram for the logic built into the Conficker.A and Conficker.B worms. (the control-flow shown at left is just another way of looking at the control-flow shown at right.)
    - creates a mutex (mutual exclusion): This will fail if there is a version of the worm already running on the machine.
    - Attach to service.exe / Attach to a running process: checks the Windows version on the machine and attaches itself to a new instance of the svchost.exe



* + - Patch dnsapi.dll: it compromises DNS lookup in machine to prevent the name lookup for organizations that provide anti-virus products.
    - worm instructs the firewall to open a randomly selected high-numbered port to the internet. It then uses this port to reach out to the network to infect other machines
    - The IP addresses chosen for infection are selected at random from an IP address database
    - rendezvous points: worm entering an infinite loop in which it constructs a set of randomly constructed 250 hostnames once every couple of hours. Since the random number generator used for this is seeded with the current date and time, we can expect all the infected machines to generate the same set of names for any given run of the domain name generation.
    - After the names are generated, the worm carries out a DNS lookup on the names to acquire the IP addresses for those 250 names. The worm then sends an HTTP request to those machines on their port 80 to see if an executable for the worm is available for download. If a new executable is downloaded and it is of more recent vintage, it replaces the old version.
  + worm have incorporated a procedure for binary code validation
    - use MD5 / MD6) to hash the binary as the encryption key
    - use RC4 to encrypt the binary with the encryption key
    - use RSA to compute digital signature and append it to the encrypted binary and together they are made available for download by hosts.
* worms have generally been programmed to attack personal computers, particularly computers running Windows operating systems, for such nefarious purposes as stealing credit-card or bank information, sending out spam, mounting coordinated denial-of-service attacks on enterprise machines, etc.
* STUXNET WORM: designed to attack industrial software SCADA to harm the processes related to the production of nuclear materials in certain countries.
  + the worm was designed to jump from personal computers to the Siemens computers used for SCADA-based process control. Once it had infiltrated SCADA, it could fake the data sent by the sensors to the central monitors so that the human operators would not suspect that anything was awry, while at the same time creating potentially destructive malfunction in the operation of the centrifuges used for uranium enrichment.
  + It caused the frequency converters used to control the centrifuge speeds to raise their frequencies to a level that would cause the centrifuges to rotate at too high a speed and to eventually self-destruct.
  + one of the three vulnerabilities exploited by the Stuxnet worm is the same as that by the Conficker work, Stuxnet worm exploits the following vulnerabilities in the Windows operation system
    - Propagation of the worm is facilitated by vulnerability related to the print spooler service in the Windows platforms. This allows the worm to spread in a network of computers that share printer services.
    - If a machine is running a prepatched version of the Windows Server Service svchost.exe and you send it a specially crafted string on its port 445, you can get the machine to download a copy of malicious code using the HTTP protocol from another previously infected machine and store it as a DLL, etc.
    - worm can propagate via removable disk drives through vulnerability in the Windows shell It allows for remote code execution if a user clicks on icon of a specially crafted shortcut that is displayed on the screen.
* WannaCry WORM
  + two modes of propagation:
    - directly exploiting a vulnerability in version 1.0 implementation of the Microsoft SMB (Samba) protocol: WannaCry hops from one host to another through random scans in which it looks for the hosts with open 445 port. It makes sure that a targeted host has not previously been infected by the same worm before installing itself in the host.
    - through the DoublePulsar backdoor if it is already installed on the host being attacked
      * DoublePulsar: exfiltrate documents from victim machine and pull in additional malicious code, such as WannaCry, from network. Remains installed in the victim host even afterWannaCry is removed. Resides only in the RAM of a host. It is memory resident malware and disappears when you reboot host.
      * Implant: a polymorphic virus that may exhibit one behavior when first installed on a host (implant may write a piece of malicious code in the boot sector of a disk) and a different behavior when the same host is subsequently rebooted (automatically execute this code and could cause the host to freeze up).
      * WannaCry does check whether DoublePulsar is already installed on the target host. If yes, it asks DoublePulsar to pull it in. Otherwise, it pushes itself into the victim machine directly through the SMB port 443 and at the same time creates the DoublePulsar backdoor on the victim host.
      * DoublePulsar can communicate with its handlers using either the RDP (Remote Desktop Protocol: transfer data between hosts) or the SMB protocol.
* alternative approaches to keep your computer from executing malware.
  + white listing: constructing a list of the DLLs that are allowed to be executed on the machine.
  + behavior blocking: uses a large number of attributes to characterize the behavior of executable code. These attributes could be measured automatically by executing the code in, say, a chroot jail on your machine so that no harm is done. Subsequently, any code could be barred from execution should its attributes turn out to be suspect.

▓The main goal of port scanning is to find out which ports are open, which are closed, and which are filtered.

* a port is filtered: packets passing through that port are subject to filtering rules of a firewall. EX. a port on a remote host is with something like an filtered iptables based packet filter and your scanner sends it a SYN packet or an ICMP ping packet, you may not get back anything at all.
* If a port on a remote host is open for incoming connection requests and you send it a SYN packet, the remote host will respond back with a SYN+ACK packet. A given port on your machine is open only if you are running a server program on the machine and the port is assigned to the server.
* If a port on a remote host is closed and your computer sends it a SYN packet, the remote host will respond back with a RST packet
* A frequent goal of port scanning is to find out if a remote host is providing a service that is vulnerable to buffer overflow attack

▓The simplest type of a scan is made with a call to connect(): int connect(int socketfd, const struct sockaddr \*address, socklen\_t address\_len);

* Socketfd: file descriptor associated with internet socket constructed by client
* Address: pointer parameter points to a sockaddr structure that contains the IP address of the remote server
* address\_len: specifies length of structure pointed to by the second argument.
* When connect() is successful, it returns the integer 0, otherwise it returns -1.
* In a typical use of connect() for port scanning, if the connection succeeds, the port scanner immediately closes the connection

▓Scanning remote hosts with SYN packets: scanner machine sends out SYN packets to the different ports of a remote machine. When the scanner machine receives a SYN+ACK packet in return for a given port, the scanner can be sure that the port on the remote machine is open. Port-scanner send back to the target machine an RST packet so that the half-open TCP circuit at the target is closed immediately.

▓nmap (network ma) Port Scanner: In addition to listing open ports on a network, it also tries to construct an inventory of all services running in a network and detect which operating system is running on each machine, etc. In addition to carrying out a TCP SYN scan, nmap can also carry out TCP connect() scans, UDP scans, ICMP scans, etc.

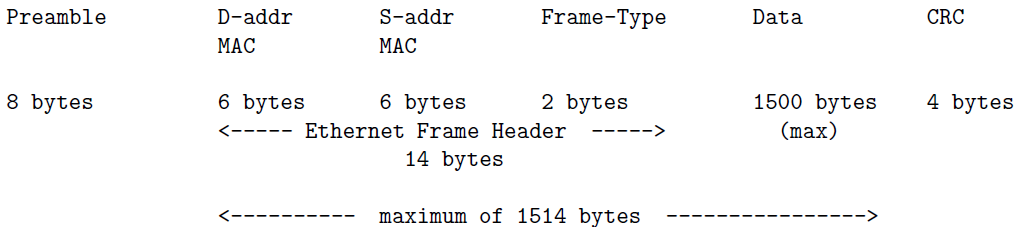
* In a UDP scan, if a UDP packet is sent to a port that is not open, the remote machine will respond with an ICMP port-unreachable message. So the absence of a returned message can be construed as a sign of an open UDP port.
* Nmap comes with a large number of options for carrying out different kinds of security scans of a network.
  + -sP : ping scanning option, ascertaining which machines are up in a network. nmap sends out ICMP echo request packets to every IP address in a network. Hosts that respond are up. But this does not always work since many sites now block echo request packets. To get around this, nmap can also send a TCP ACK packet to (by default) port 80. If the remote machine responds with a RST back, then that machine is up. Another possibility is to send the remote machine a SYN packet and wait for an RST or a SYN/ACK. For root users, nmap uses both the ICMP and ACK techniques in parallel. For non-root users, only the TCP connect() method is used.
  + -sV : Version Detection. A file called nmap-services-probes is used to determine the best probes for detecting various services. In addition to determine the service protocol (http, ftp, ssh, telnet, etc.), nmap also tries to determine the application name (such as Apache httpd, ISC bind, Solaris telnetd, etc.), version number, etc.
  + -sT : TCP connect() scan. Port scanning with calls to connect()
  + -sU : sends a dataless UDP header to every port, the state of the port is inferred from the ICMP response packet (if there is such a response at all).
  + nmap -sS -A moonshine.ecn.purdue.edu: carry out an aggressive SYN scan of moonshine.ecn.purdue.edu, enables OS detection, version scanning, script scanning, and more. -sS option carries out a SYN scan, -A option as standing for either aggressive or advanced. If the target machine has the DenyHosts shield running to ward off the dictionary attacks and you repeatedly scan that machine with the -A option turned on, your IP address may become quarantined on the target machine. When that happens, you will not be able to SSH into the target machine.
  + limit the range of ports to scan with the -p option: only the first 1024 ports to be scanned: nmap -p 1-1024 -sT moonshine.ecn.purdue.edu
  + The larger the number of router/gateway boundaries that need to be crossed, the less reliable the results returned by nmap.
  + By default, nmap first pings a remote host in a network before scanning the host. But since many sites now block/filter the ping echo request packets, this strategy may bypass machines that may otherwise be up in a network. -P0 option tells nmap to not use ping in order to decide whether a machine is up. (nmap -sS -A -P0 moonshine.ecn.purdue.edu)
  + Nmap can make a good guess of OS running on target machine by using TCP/IP stack fingerprinting. It sends out a series of TCP and UDP packets to target machine and examines content of returned packets for the values in the various header fields. Based on these values, Nmap then constructs an OS signature of the target machine and sends it to a database of such signatures to make a guess about the OS running on the target machine.

▓security scanner / vulnerability scanner / security vulnerability scanner: scans a specified set of ports on a remote host and tries to test the service offered at each port for its known vulnerabilities. Frequently include port scanning.

* It is a scanners job to connect to all possible services on all the open ports on a host. If the TCP engine on the machine is poorly written, the machine may get overwhelmed by the network demands created by the scanner and could simply crash. Thats why many sysadmins carry out security scans of their networks no more than once a month or even once a quarter.
* Nessus Vulnerability Scanner: a remote security scanner, typically run on one machine to scan all the services offered by a remote machine to determine whether remote machine is safeguarded against all known security exploits.
* Nessus vulnerability scanning system consists of a server and a client.
  + server program is called nessusd: It attacks other machines in a network, typically installed at the path /opt/nessus/sbin/nessus
  + client program is called nessus: at the path /opt/nessus/bin/nessus, it tells the server as to what forms of attacks to launch, what category of plugins to use, and where to deposit the collected security information.
* The security tests for the Nessus system are written in a special scripting language called Network Attack Scripting Language (NASL). Each security test, written in NASL, consists of an external plugin. New plugins are created as new security vulnerabilities are discovered. The command nessus-update-plugins can automatically update the database of plugins on your computer
* Nessus can detect services even when they are running on ports other than the standard ports and apply the applicable tests at those ports. (HTTP service is running at a port other than 80)
* Nessus can test SSLized services such as HTTPS,SMTPS, IMAPS, etc.

▓packet sniffer / network analyzers / analyzers protocol: is a passive device

* network analyzer: you can use a packet sniffer to localize a problem in a network. If network interface on a particular host is not seeing packets, you can be a bit more certain that problem may be with network interface in question.
* protocol analyzer: a packet sniffer can look inside the packets for a given service and make sure that the packet composition is as specified in the RFC document for that service protocol.
* What makes packet sniffing such a potent tool is that a majority of LANs are based on the shared Ethernet. So all the Ethernet interfaces on all the machines that are plugged into the same router will see all the packets. On wireless LANs, all the interfaces on the same channel see all the packets meant for all of the hosts that have signed up for that channel.
* Physical Layer puts information (consists of data packets called frames) on wire. Each Ethernet interface gets a 48-bit MAC address to specify the source address and the destination address in each frame. Even though each network interface in a LAN sees all the frames, any given interface normally would not accept a frame unless the destination MAC address corresponds to the interface.
* structure of an Ethernet frame:



* + Preamble: announce the arrival of a new frame and to enable all receivers in a network to synchronize themselves to the incoming frame.
  + D-addr stands for destination address
  + S-addr for source address
  + Type: identifies the higher-level protocol (e.g., IP or ARP) contained in the data field, tells us how to interpret the data field.
  + CRC (Cyclic Redundancy Check): provides a mechanism for the detection of errors that might have occurred during transmission. If an error is detected, the frame is simply dropped.
* From the perspective of a packet sniffer, each Ethernet frame consists of a maximum of 1514 bytes (plus 4 bytes CRC checksum is 1518). Minimum size of an Ethernet frame is 64 bytes. Padding bytes must be added if the data itself consists of fewer than 46 bytes.
* Data Link Layers map the destination IP address in an outgoing packet to the destination MAC address and to insert the MAC address in the outgoing frame. The Physical Layer then puts the frame on the wire.
* Why not use the IP addresses directly as MAC addresses for communications in a local network? That would not be practical since we must allow a host to possess multiple communication interfaces, especially for a router. With the clean separation between IP addresses and MAC addresses, a single host with a unique IP address is allowed to have an arbitrary number of interfaces, each with its own MAC address.
* Data Link Layer uses protocol Address Resolution Protocol (ARP) to figure out the destination MAC address corresponding to the destination IP address. System looks into the locally available ARP cache. If no MAC entry is found in this cache, the system broadcasts an ARP request for the needed MAC address.
* When a network interface does not discriminate between the incoming frames on the basis of the destination MAC address, we say the interface is operating in the promiscuous mode.
* Dsniff packet sniffer contains the following utilities:
  + Sshmitm: can launch a man-in-the-middle attack on an SSH link. It intercepts the public keys being exchanged between two parties A and B wanting to establish an SSH connection. The attacker, X, eavesdrop on the communication between A and B with the help of a packet sniffer pretends to be B vis-à-vis A and A vis-a-vis B.
  + urlsnarf : From the sniffed packets, this utility extracts the URLs of all the web sites that the network users are visiting.
  + mailsnarf: track all the emails that the network users are receiving.
  + webspy : track a designated users web surfing pattern in real-time.

▓snort: an intrusion detector, bring to bear on the packets some fairly complex logic to decide whether an intrusion has taken place.

* alert tcp any any -> 192.168.1.0/24 80 (content:"|A1 CC 35 87|"; msg:"accessing port 80 on local")
  + rule header.
    - alert: the action part of the rule,
    - tcp: the protocol part,
    - any any: the source address and the source port
    - ->: the direction operator, direction operator can be either ->, <-, <>, the last for packets going in either direction.
    - 192.168.1.0/24 80: the destination address and port
  + inside () is the rule body: rule options separated by ;
    - payload detection option: establish a criterion for triggering the rule
      * content: looks for a string of bytes in the packet payload
      * nocase: that makes payload detection case insensitive.
      * Offset: specifies how many bytes to skip before searching for the triggering condition,
      * Pcre: matching of the payload will be with a Perl compatible regular expression,
    - metadata option: convey useful information back to human operator.

▓Metasploit Framework can be thought of as a major force multiplier for both the good guys and the bad guys. It makes it easier for the good guys to test the defenses of a computer system against a large array of exploits that install malware in your machine. At the same time, the Framework makes it much easier for the bad guys to experiment with different exploits to break into a computer.

* MF command msfpayload: allows you to create a payload in either the source-code form in a large variety of languages or as a binary executable for a number of different platforms.
* Metasploit Framework creates two different kinds of payloads:
  + Payloads that are fully autonomous for whatever it is they are meant to do
  + Payloads with just sufficient networking capability to later pull in the rest of the needed code.
* false-positive rate of an anti-virus tool that detects the second type of a payload would generally be much too high for the tool to be of much practical use.
* From standpoint of good guys, a payload is what you attack a machine with to test its defenses. And, from the standpoint of the bad guys, a payload is a worm

▓dictionary attack: most commonly used ploy to break into port 22, try a large number of commonly used names as possible account names on the target machine and if there is actually an account on the target machine, they then try a large number of commonly used passwords for that account.

▓log scanning: protect a computer/network against a dictionary attack

◆Fail2Ban: detect intrusion attempts by keeping track of the number of login attempts. It is the most commonly used tool for intrusion prevention through log scanning. By using regex based filters, Fail2Ban can try to detect malicious behaviors by connections made by IP addresses (say, for downloading web pages) and subsequently it can take any action you wish vis-a-vis those addresses. IP

* there is a fundamental difference in how the two tools keep the blacklisted IP addresses at bay: (with both tools, no further SSH connections from the same IP address would be honored at least until expiration of a certain time interval)
  + Fail2Ban: blacklisted IP address is kept out by adding a new rule to the iptables firewall.
  + DenyHosts: places a blacklisted IP address in the file /etc/hosts.deny
* With log scanning of the sort used in Fail2Ban and DenyHosts, your security decision is based more on actions of a clumsy thief who is unsuccessful and not on actions of those who may have caused you serious harm in the past. Even a successful thief may need to make a few attempts before hitting the jackpot,
* DenyHost was created exclusively for monitoring the SSHD access log files.
* Fail2Ban
  + one of the best things about Fail2Ban is its versatility: It can block network access to just about any application that creates a log file for incoming connection requests. (list in /etc/fail2ban/jail.conf). Even for the same server application running in your computer, it can identity IP addresses that are engaged in different malicious activities and, depending on what activity is involved, it can take different actions. Fail2Ban detects activities with the help of filters based on regular expressions.
  + Parameters in /etc/fail2ban/jail.conf
    - Bantime: specifies in seconds the duration of time for which a blacklisted IP address is denied further access.
    - Findtime and maxretry: If intruder makes more than maxretry attempts during a findtime period, IP address is quarantined for the duration set by bantime
    - Mta: specifies mail transport agent to use for sending an email notification to a designated person/admin when an IP address is blacklisted.
    - Destemail: notification is sent to account specified by destemail
    - Action: tells Fail2Ban what to do with an IP address that meets the repeat access conditions as set by the findtime and the maxtry parameters. With the action\_mw choice, the email notification will include a whois report on the intruding host. And, with action\_mwl, the email notification will include relevant log lines.
* DenyHosts
  + in addition to entering a blacklisted IP address in in the /etc/hosts.deny file, the blacklisted IP addresses are also recorded in in a few more files elsewhere in your directory system for synchronizing your blacklisted IP addresses with similar such addresses collected by other hosts in the internet if you have the synchronization option turned on in the config files
  + main config file for DenyHosts is /etc/denyhosts.conf.
    - DENY\_THRESHOLD\_INVALID: the limit on how many times an intruder can try to gain entry with usernames that do NOT exist in the /etc/passwd file
    - DENY\_THRESHOLD\_VALID: limit on trying to gain entry through usernames that actually do exist.
  + DenyHosts makes its log entries in the file /var/log/denyhosts

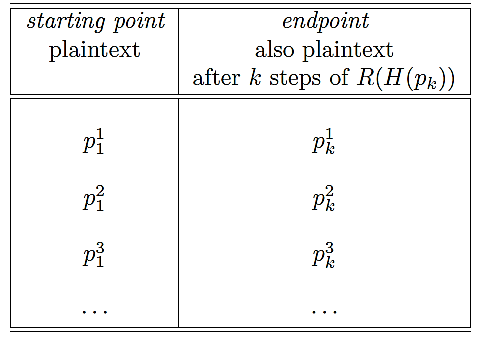
▓Password cracking: you have already broken into a machine and somehow gotten hold of the document (System Password File) where all the password hashes are stored. Now you want to map the password hashes back to the character strings that are the passwords as entered by the users.

◆two facts have given much impetus to the development of password cracking methods during the last twenty years

* ubiquity of the Windows machines all around the world.
* older versions of the Microsoft Windows platform used an extremely weak method for hashing passwords: LM (LAN Manager) Hash, ASCII string for password can be inferred from its hash value through rainbow table attack
  + LM Hash algorithm:
    - password is limited to a maximum of 14 ASCII characters and zero-padded to 14 if shorter than that.
    - lowercase characters in password are converted to uppercase.
    - 14-character string is divided into two 7-character substrings, each substring used as a key to the DES algorithm to encrypt the 8-character plaintext string KGS!@#$%.
    - Each half produces a 64-bit ciphertext and two ciphertext bit streams are simply concatenated together to create a 128-bit pattern that is stored
  + vulnerabilities that are specific to the LM Hash
    - it is easy to guess if the original password string was shorter than 8 characters since in all such cases the second half the input string is all zeros and it results in the predictable DES encryption given by the hex 0xAAD3B435B51404EE.
    - Hash is that the two halves of the hash value can be attacked separately since there were calculated independently.
    - since LM Hash converts lowercase to uppercase, each character can only be one of 69 values (not 95). Therefore, the total number of distinct hash values for each 7-character part of the password is 69^7. Total number of password strings of length 7 or less is given by 69^7 + 69^6 + 69^5 + 69^4 + 69^3 + 69^2 + 69.
  + hash-table database: construct a hash for all possible character combinations and to then store these <password, hash> values
  + lookup-table attack: password hash is attacked by looking up a table of previously computed hashes,
  + hash chain: hash chains requires reduced memory but at the cost of having to spend more time to get to the password
    - reduction function R: Any mapping that more or less uniformly samples the space of all possible passwords is a good enough mapping. Reduction function may map more than one hash to the same password.
    - c = H(p), p’= R(c), Given the pair of functions H() and R(), randomly chosen plaintext p1 from the space of all passwords, construct a hash chain which length is k



and store it as a table



* + test hash chain



If any of q1, q2, … match any of the endpoints in the second column of the table shown above, then there is a high probability that the password that the cracker is looking for is in the chain corresponding to that row.

* + The cracker will now regenerate the chain for the ith row of the table.



With a significant probability, the cracker will find that his hash C matches one of the hashes in this chain. Once a match is found, the password that the cracker is looking for is the plaintext that immediately precedes C in the chain.

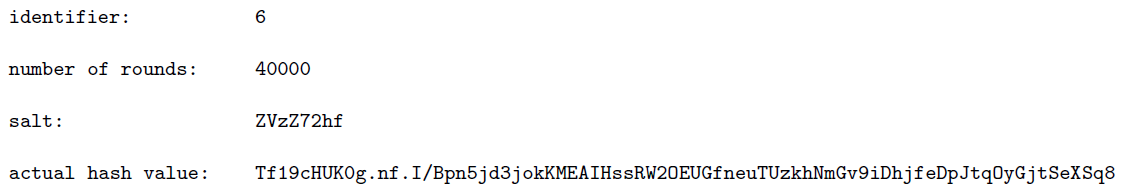
* + as we grow the test hash chain one step at a time starting with the hash C to be cracked, we run into a qm that matches one of the endpoints in our table, but we are unable to find C in the chain for that row. In such an event, we continue to grow the test chain and look for another qn that matches but we do NOT grow the test hash chain beyond the k steps.
  + hash chain table should have property that passwords stored implicitly in all the chains should span the space of all possible passwords.
  + Chain 1 contains a specific password at step i and Chain 2 has the same password at step j. The endpoints will be different because number of remaining steps in the two chains is not the same.
  + Collision: When two different chains in a table show significant overlap. This cannot be detected because we only store the starting points and the endpoints for the chains.
  + such implicit overlaps can significantly reduce the ability of a hash chain table to crack a hash because of the reduced overall sampling of the space of all the passwords.
  + Merging of the chains (overlap between the hash chains) places an upperbound on the size of a hash chain table.
  + construct a hash chain table for a large number of randomly selected starting points in the space of all passwords. But, as the size of the table grows, the table becomes more and more inefficient on account of chain merging. Before the invention of rainbow tables, this problem was taken care of by constructing a number of hash chain tables, each with a different reduction function R().
  + rainbow tables
    - construct a single hash chain table, but use k different reduction functions for each of the k steps in the construction of a chain.
    - For a collision to occur, the password that is reduced to must be the output of the same reduction function an event with much lower probability than was the case with hash-chain tables as presented above.
    - in hash-chain tables, there is always a possibility that you will encounter a loop as you grow a chain. Since a reduction function is intentionally many-to-one, there is always a chance that the password that is reduced to will be the same at two different places in a chain. However, when you use different reduction functions for the successive reduction steps in a chain, you are less likely to run into loops.
    - lookup consists of first applying the last of the reduction functions R\_k() to obtain q1= R\_k(C) and then checking whether q1 is an endpoint in the rainbow table. If not, we grow the test chain by calculating q2 = R\_k-1(H(q1)) and search for q2 as an endpoint in the table. If a matching endpoint cannot be found for q2, we grow the test chain by one more step by calculating q3= R\_k-2(H(q2));and so on.

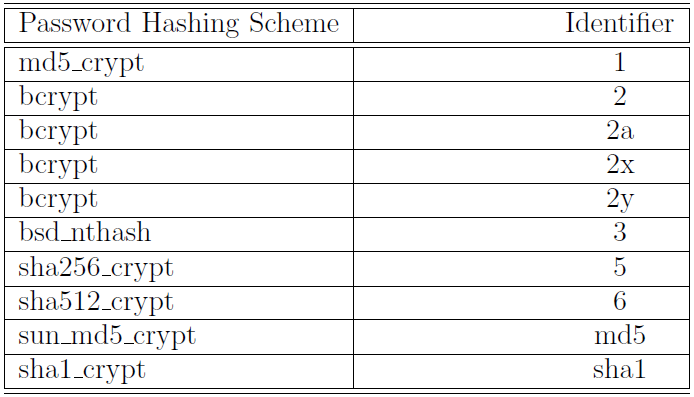
▓reason why you cannot just directly apply an algorithm such as SHA-512 to a user-entered password string is because the resulting hash values would still be crackable despite the fact that hash function itself is cryptographically secure and possesses the one-way property (if the password only has 6 lowercase characters, total number of possibilities is only 26^6). Therefore, all modern password hashing schemes

combine with the user-chosen password string a number of random bits (salt)

▓MCF (Modular Crypt Format) is the format for storing password hashes in /etc/shadow

* $<identifier>$rounds=<number-of-rounds>$<salt>$<password-hash>
* $6$rounds=40000$ZVzZ72hf$Tf19cHUK0g.nf.I/Bpn5jd3jokKMEAIHssRW2OEUGfneuTUzkhNmGv9iDhjfeDpJtqOyGjtSeXSq8





1. Identifier: the Password Hashing Scheme.
2. Rounds: number of rounds applied by the password hashing scheme to hash a password. By hashing a multiple number of times, you make it that much harder to crack a password through any sort of a table lookup, rainbow or otherwise, especially if the number of rounds is randomly chosen for each user account.
3. A salt is a randomly chosen bit pattern that is combined with the actual password before it is hashed by a hashing algorithm. (ZVzZ72hf in he example). These are eight Base64 characters, each standing for six bits. Even intruder who stole the file /etc/shadow knows the salt used for each username. Nonetheless, he/she would not be able to use precomputed rainbow tables available on the web, it would simply take much too long (possibly years) for the intruder to create his/her own rainbow tables that accounts for every possible value of salt.
4. a side benefit of using a random value for salt is that it makes less likely that any two usernames will have the same password hash associated with them.
5. Base64 encodings as used by password hashing schemes are also known as Hash64 encodings.
6. hashing algorithm of password hashing scheme sha512 crypt
   1. creates multiple replications of a concatenation of the user-supplied password string, the salt, followed again by the password string, the number of such concatenations used being the number 64-byte blocks in the original password string (with provision for the
   2. print passlib.hash.sha512\_crypt.encrypt(avikak, rounds=5000, salt="ZVzZ72hf")

▓Bayesian filters: statistical filter with sufficiently low falses would require too many samples of a certain type of spam before blocking such messages in the future.

▓regular-expression based filter: once you see a spam message that has leaked through, you can design a short regular expression to block the email you just saw and all its variations that the spammer may use in the future in just one single step.

▓MTA (Mail Transfer Agent) / Mail Transport Agent / Mail Exchange Server: is used to transfer email to another MTA in the internet. The main function of an MTA is to exchange email with another MTA, they can also be programmed to receive email directly from MUAs and to send messages directly to the same.

* the client email first goes to an MSA (Mail Submission Agent) and the MSA forwards it to the MTA. MTA forwards the email to an MDA (Mail Delivery Agent). MDA send that email to the clients.
* MTA can also be programmed to send email directly to the clients.

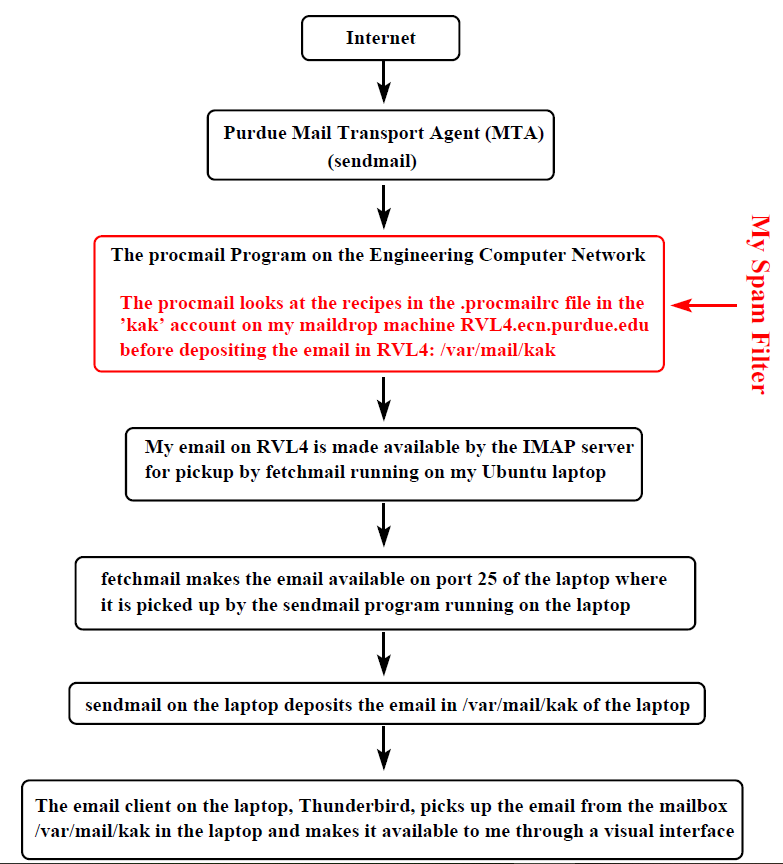
▓send an email to [kak@purdue.edu](mailto:kak@purdue.edu)

1. name resolver associated with the email client being used by the sender will ask the DNS servers for the IP address of the host that is designated to be the mail exchange server for the purdue.edu domain.
2. MTA program running on this host at Purdue receive the email sent to me.
3. MTA may use either a Mail Delivery Agent (MDA) to deliver a received email to recipients mailbox, or deliver it directly to the recipients mailbox.
4. MDAs main job is to apply appropriate filters to email before it is deposited in the mailboxes of the user accounts. These filters may be at the system level, in which case they can affect all users, or at the level of individual users. Filters used by MDA take the form of recipes that are placed in a file named .procmailrc. The .procmailrc file for filters that are specific to individual users must reside at the top level of the user-account home directories.
5. After the email is deposited in a user mailbox, it may be read by the user with the help of an MUA (Mail User Agent), also called email client.
6. read my email:

ssh kak@rvl4.ecn.purdue.edu

tail -f Mail/logfile

local email exchange server sends email to machine rvl4.ecn.purdue.edu. “tail -f” shows the latest entries created by Procmail in the Mail/logfile. Logfile that you see is created by my Procmail spam filter.



▓An email consists of three parts:

* 1. body: carries the message of the email, may contain multimedia objects.
  2. header: Contains the From:, To:, Cc:, etc., information. It does NOT usually tell you the route the email took from the sender to the recipient. The header of an email message ends at the first empty line encountered from the top. What comes after that empty line is the body of the email. [It is important to know where exactly the header of an email ends and where the body begins. That is because spam filter rules can be based on just the header, or just the body, or both.]
  3. envelope: This part is usually suppressed by an MUA. [Some MUAs provide you with a menu option to see all the headers, including the routing headers.] It consists of the conversation that takes place between a sender MTA and a receiver MTA involving recipient authentication, etc.

▓For an email to be recognized as legal by an MTA, its very first line must begin with From. There can be no punctuation marks attached to this word. In other words, it

can only be followed by a space. To make sure that MTAs are not confused by From at the beginning of the body, an MSA typically prefixes such a From with the character >. Such a problem does not arise if you have asked your email client to send messages formatted according to, say, HTML.

▓It is the recipients name in the envelope of an email that determines where an email ends up and NOT what shows up in To: in the header part of an email.

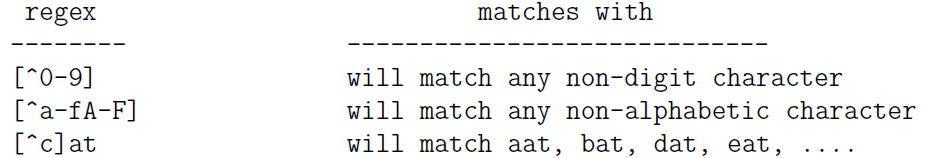
▓MUA may modify the very first From line into two separate lines, one for Return-Path and the other for Delivery-Date. So what an MTA sends an MDA may not be the same as what the MUA stores for the email and that, in turn, may not be the same as what the MUA actually shows you on the screen.

▓anchor metacharacters

* ^: is used to force a match to take place at the beginning of the input string
* $: force the match to take place at the end of the input string.
* \b: stand for a non-word to word transition and a word to non-word transition.

▓character class is simply a set of choices available for a specific character position in a regex. EX. s[tpk]ool for stool, spool, skool

* character-class metacharacters: - ^ ] \
  + “-“ loses its special meaning if it is either the first or the last character inside the square brackets. [0-9a-fA-F]: a digit or letter in a hex sequence
  + If ^ is the first character inside square brackets, it negates the entire character class. If ^ appears anywhere except at the beginning of a character class, it loses its special meaning vis-a-vis the character class.



* “|”specifying alternative subexpressions in a regex. EX. \b(Jo(e|seph))|Mary\b. The regex engine seeks the earliest possible match and, as soon as the engine is successful, stops trying out any remaining alternatives
* Grouping metacharacters () groups in a regular expression. Subexpressions34