* DoS attacks with TCP

1. Shrew DoS (Denial of Service) attack: hitting a host periodically with a square wave of short duration DoS, bring down a TCP engine.

* + To guarantee reliability in communications, TCP retransmits a TCP segment when

(1) an ACK is not received in a certain period of time;

(2) three duplicate ACKs are received consecutively

* + The sender TCP detects congestion by non-arrival of an ACK packet within a dynamically changing time window or by the arrival of three consecutive duplicate ACK packets. Retransmission decision for a TCP segment is based on two different timescales:

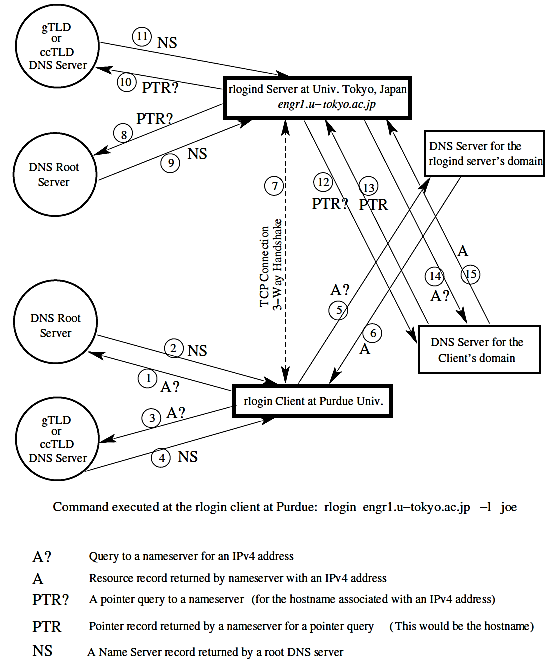
(1) RTT (Round Trip Time): When traffic congestion is low, AIMD

(2) RTO (Retransmission Timeout): when congestion is high. When no ACK is received within an RTO, that indicates severe congestion. Initial value of RTO depends on RTT. When RTT cannot be measured, the initial value for RTO value is set to 3 sec, the minimum being 1 sec.

* + sender TCP engages in congestion control by changing value in its CWND field.【controls the size (in units of SMSS Sender Maximum Segment Size) of TCP segment that is sent to IP Layer】
  + AIMD (Additive Increase Multiplicative Decrease) is used to set values in the CWND field.
    - sender TCP sends out a TCP segment whose size is the starting value for CWND, which is one MSS.
    - If an ACK within an RTT, the sender sets the value CWND = CWND + 1 SMSS. If ACK packs keep coming back within one RTT, the size of the transmitted TCP segment keeps on increasing linearly.
    - If an ACK not be received within an RTT, CWND = CWND\*b. b may be a fraction like 1/2. EX.
  + RTO is used for congestion control at the RTO timescale:
    - If an ACK is not received within the currently set value for RTO, then the value placed in the CWND window is reduced to 1.
    - If an ACK is not received again, the RTO is doubled, while the CWND value maintained at 1.
    - if an ACK is received within the currently set RTT, TCP switches back to the RTT timescale logic for congestion control.
  + The manner in which RTO is set and reset can be exploited to Shrew DoS (Denial of Service) attack on a sender TCP.
    - timeout mechanism for robust congestion control, provides an opportunity for low-rate DoS attacks
    - an attacker can provoke a TCP flow to repeatedly enter a retransmission timeout state by sending a high-rate but RTT-scale length bursts and repeating periodically at slower RTO timescales.
    - The victim will be throttled to near zero throughput, while the attacker will have low average rate making it difficult for counter-DoS to detect
    - single TCP flow case:
      * RTO at the sending TCP targeted by attacker is 1 sec. The attacker will send TCP with a short burst of DoS packets. The duration of this burst is RTT for the communication link that attacker wants to bring down.
      * host reset its RTO to 1 second and CWND to 1 SMSS. In response to the congestion, the sending TCP will send out one packet of length CWND and wait for the RTO of 1 sec for an ACK.
      * attacker send another DoS burst at the end of 1 sec, the sending TCP will double RTO to 2 seconds while keeping CWND at 1.
      * If the attacker persists in hitting the victim TCP with these DoS bursts at every new value of RTO, TCP flow emanating from victim machine would virtually halt.
  + shrew DoS attack is more difficult to detect because it would not be detectable by a traffic monitor that is looking for heavy traffic associated with the more common DoS attacks.

2. TCP SYN flooding:

* + TCP SYN flood attack on a server host:
    - hostile client repeatedly sends a TCP SYN request to every port on the server using a fake IP address.
    - server responds to each with a SYN/ACK response from each open port and an RST response from each closed port.
    - In a normal three-way handshake, the client would return an ACK packet for each SYN/ACK packet received from the server. However, in a SYN flood attack, the hostile client never sends back the expected ACK packet. And as soon as a connection for a given port gets timed out, another SYN request arrives for the same port from the hostile client.
    - When a connection for a given port at the server gets into this state of receiving a never-ending stream of SYN requests, intruder has a half-open connection with the victim host.
  + server can modify its firewall rules so that all SYN packets arriving from the intruder will be simply discarded or protect its resources by rate limiting all incoming SYN packets.
  + SYN scanning.: The transmission by a hostile client of SYN connection requests for the purpose of finding open ports
  + IP source address spoofing: an intruder using one or more forged source IP addresses to launch a TCP SYN flood attack. As soon as the attack is detected, the admins of the targeted network will block the source IP addresses. Such packet filtering would amount to a denial of service (DoS) to the legitimate users/systems at those IP addresses.
  + the attacker may not only cause a denial of service at the forged IP addresses, but may also cause SYN/ACK flooding at the victim hosts which slow down the performance of routers for handling the legitimate traffic or cause them to crash.
  + DoS attack through IP address spoofing is becoming difficult to launch. ISPs that have implemented RFC 2827 do not allow their routers to send out packets if their source IP address does not fall in the range assigned to the ISP. Each router, the gateway of a LAN to the rest of the internet, works with an assigned range of IP addresses that are stored in its routing table. If a packet appears at a router whose source IP address is at odds with the routing table in the router, the packet would be discarded.
  + how IP spoofing works
    - two hosts A and B and another host X controlled by an adversary. B runs a server program that allows A to execute commands remotely at B. A and X are on the same LAN. For the attack to work, X has to pretend to be A. (source IP address on the outgoing packets from X must appear to come from A as far as B is concerned.)
    - X posing as A sends a SYN packet to B with a random sequence number M
    - Host B responds back to A with a SYN/ACK packet: (sequence num : N, acknowledgment num : M+1)
    - X at the same time suppress A’s ability to communicate with B by mounting a SYN flood attack on A. By sending a number of packets to A just prior to attacking B. A will get filled up with connection-setup requests, so the login ports of A will not be able to send to B any RST packets in response to the SYN/ACK packets that A will receive
    - To guess the sequence number that B will use, X need to gain some insights into B’s Initial Sequence Number (ISN) generator. X sends B a number of connection-request packets without posing as any other party. When B responds to X with SYN/ACK packets, X sends RST packets back to B. In this manner, X receive a number of sequential outputs of B’s random-number generator without compromising Bs ability to receive future requests for connection.
    - X will not see return from B since the routers will send it directly to A. X (posing as A) sends an ACK packet to B with a guessed value for the acknowledgment number N + 1.
    - X will have a one-way connection with B and able to send commands to B. B could execute these commands assuming they were sent by the trusted host A.
* DNS (Domain Name Service / Server / System / Space): translate symbolic hostnames into IP addresses and vice versa. Lists mail exchange servers that accept email for different domains.
  + 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.
  + DNS hijacking: malware may overwrite 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
  + /etc/host.conf tells the system in what order it should search through the two sources of hostnames-to-ipaddress mappings: /etc/hosts and DNS
  + DNS system is organized in a hierarchical fashion: top of the hierarchy are 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.
    - Generic Top Level Domain DNS server (gTLD): If root server receives a query for .com domain, root server sends back IP address of one or more gTLD in charge of .com domain.
    - Country Code Top Level Domain DNS server (ccTLD): if a root server receives a query for .jp domain, root response back the IP address of the ccTLD in charge of the .jp domain.
  + login connection with a remote host: connection that involves no exchange of security information (Messages1~6 constitute iterative namelookup for the numerical IP address associated with a domain namenor a hostnname.)

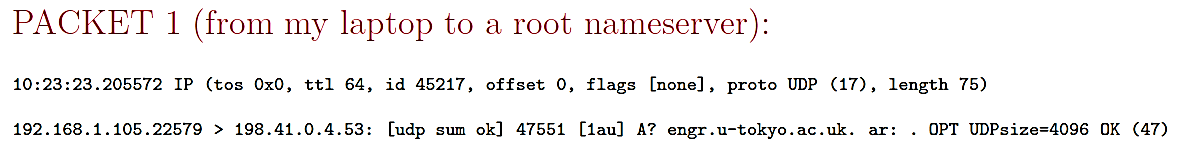


1. Client

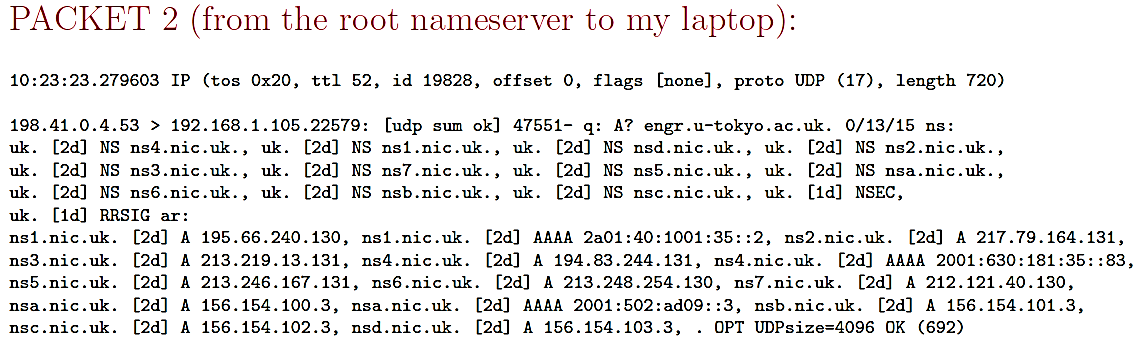
* + - client enters the rlogin command. Client resolver contacts one of the root nameservers for where to go for resolving the names .jp
    - root nameserver responds back with the IP address of the ccTLD DNS server in charge of the top-level .jp domain.
    - client contact ccTLD nameserver for the .jp domain.
    - DNS server responds back with the IP address for the authoritative nameserver for the /u-tokyo.ac.jp domain.
    - client contact the nameserver for the u-tokyo.ac.jp domain.
    - nameserver responds back with the desired IP address.
    - client TCP send a SYN packet to server for initiating connection.
    - server engages in a 3-way handshake to complete a TCP circuit.

2. server

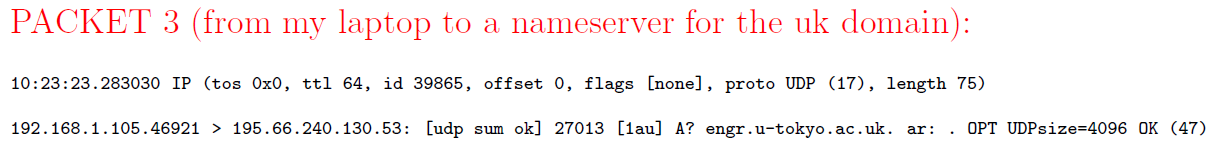
* + - rlogind server wants to know hostname of client that connected with it, so it sends a pointer query to one of the root servers.
    - root nameserver responds with IP address of gTLD or ccTLD nameserver that is relevant to the IP address in the pointer query.
    - rlogind server contacting ccTLD nameserver for in-addr.arpa domain relevant to the IP address in question.
    - DNS server responds with the IP address for authoritative nameserver for more specific in-addr.arpa nameserver relevant to the pointer query.
    - rlogind server sends the same pointer request to domain-specific nameserver.
    - server obtains fully qualified domain name (FQDN) of the client.
    - to account for the possibility that nameserver for the in-addr.arpa domain 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.
    - DNS server for client’s domain supplies the IP address associated with symbolic hostname for the client.
    - rlogind server compares this IP address with IP address in TCP connection. If IP addresses are the same, server allows connection
  + DNS request for name lookup is sent in UDP packet. 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 hostname given to ssh command does NOT exist



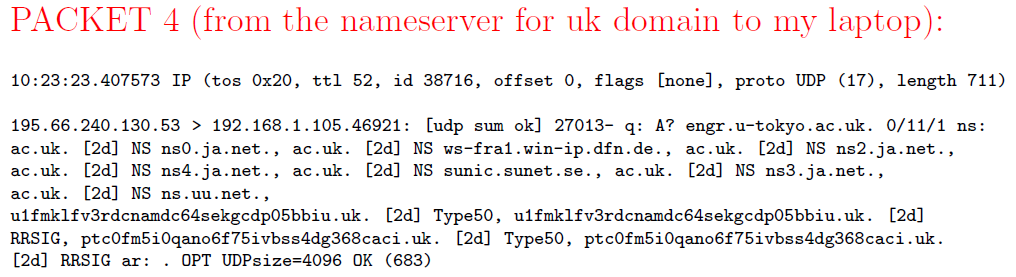
* + - “192.168.1.105.22579 > 198.41.0.4.53” says that my laptop, whose IP address is 192.168.1.105, is using port 22579 to send a UDP packet to the root server whose IP address is at its port 53.
    - Transaction ID of a DNS query 47551 is a randomly generated integer, making it difficult to mount a DNS cache poisoning attack. A valid answer to a DNS query must contain the same integer.
    - “A? engr.u-tokyo.ac.uk” means that laptop is requesting the IPv4 address for the hostname engr.u-tokyo.ac.uk.



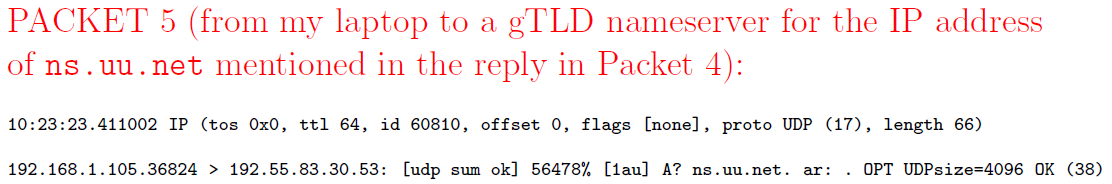
* + - The answer returned by root nameserver consists of the symbolic names and IPv4 addresses for several nameservers responsible for the uk domain. For example, one of the nameservers listed for the uk domain is ns1.nic.uk and its IPv4 address is 195.66.240.130
    - “ns1.nic.uk.[2d] A195.66.240.130”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

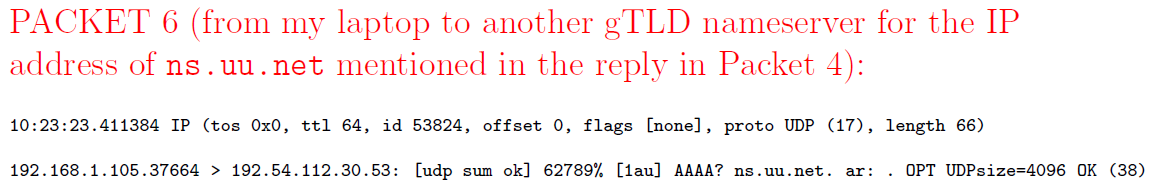


* + - “192.168.1.105.46921 > 195.66.240.130.53” tells that this is a packet from my laptop to the nameserver for the ns1.nic.uk uk top-level domain. Transaction ID number is 27013

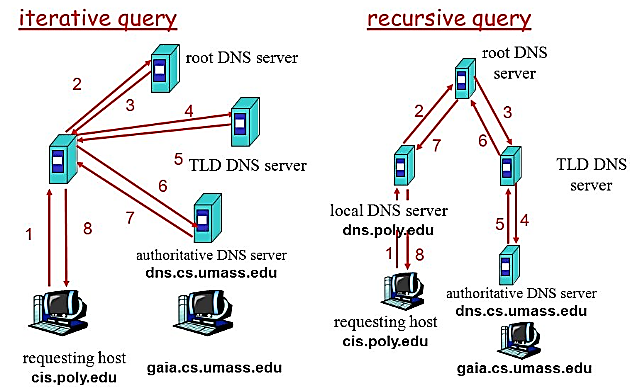


* + - nameserver responds back by sending to my laptop the symbolic hostnames for several nameservers for the ac.uk subdomain.





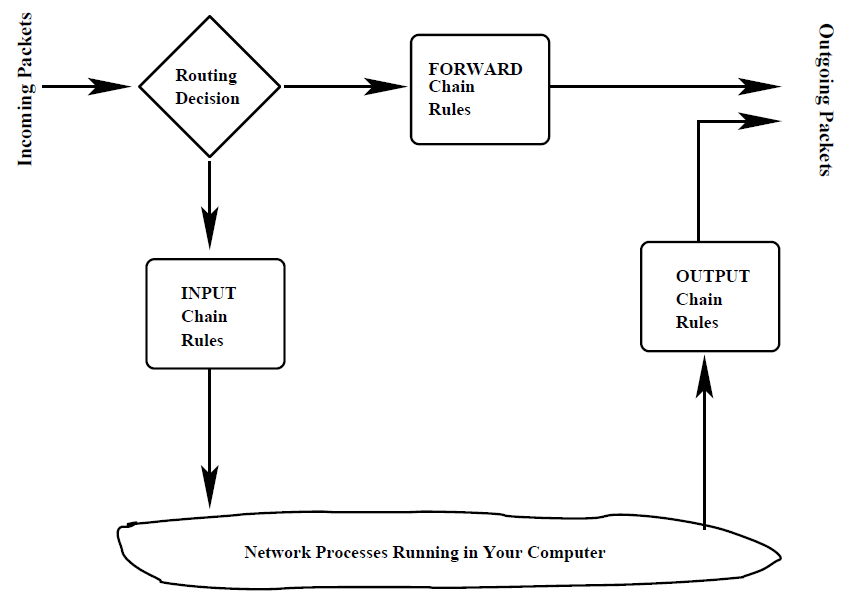
* + - nameserver on my laptop figures out there is no IP address to be had for the hostname engr.u-tokyo.ac.uk
  + each host is represented in DNS by two DNS records: an address record and a reverse mapping pointer record.
  + fully qualified domain names (FQDN): The root domain is represented by a period ”.” and formal DNS name of the purdue.edu domain is purdue.edu. period at the end stands for the root of the DNS tree.



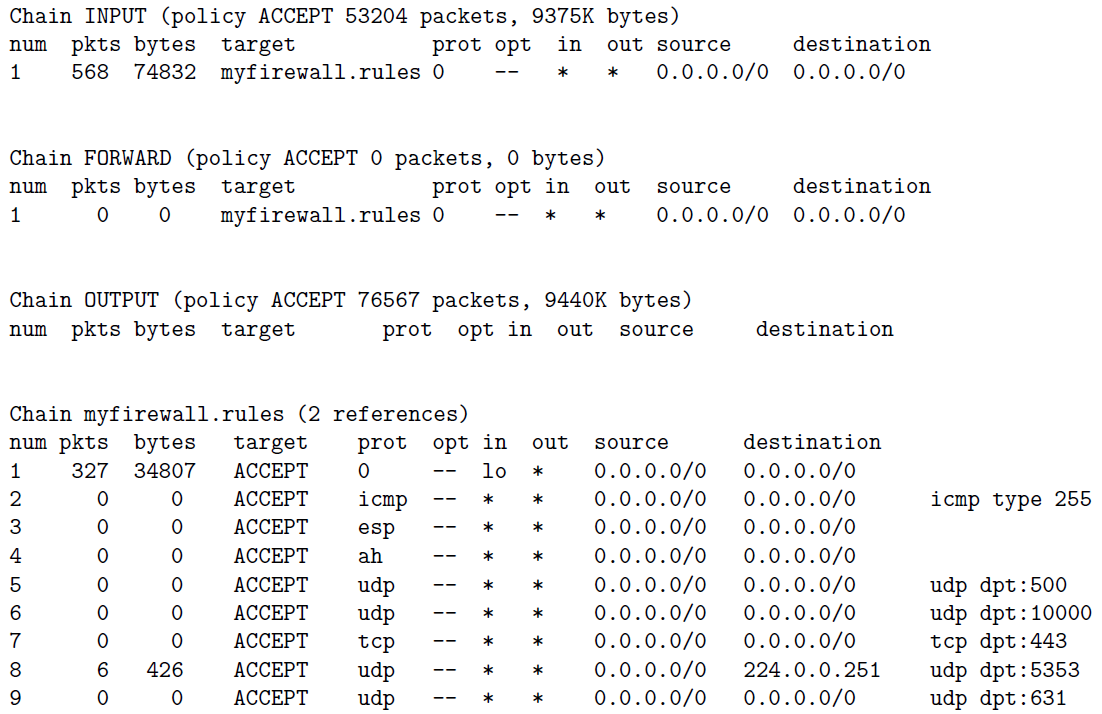
* + For the Domain Name System, the internet is divided into a tree of zones. Each zone, consisting of a Domain Name Space served by a DNS nameserver that consists of two parts:
    - an Authoritative Nameserver for IP addresses for which the zone nameserver directly knows the hostname-to-IP address mappings;
    - a Recursive Nameserver for all other IP addresse
  + zone file: authoritative nameserver file that contains the mappings between the hostnames and the IP addresses
  + DNS CACHE
    - 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.
* It is a client application such as Internet Explorer, Firefox, a mail client such as sendmail, etc., that sends a query to a DNS nameserver. 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. Operating system would be programmed to look up information in /etc/hosts for any direct hostname-to-IP address mappings you might have placed there.
  + TTL Time Interval
    - TTL value associated with a hostname is set by administrator of authoritative DNS server that returns IP address along with its TTL.
    - While DNS caching makes hostname resolution faster, any changes to the DNS do not always take effect immediately and globally.
    - making the DNS system secure against large-scale DoS attacks
      * even if root servers were taken down, 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 recovering root servers.
      * if an adversary took down the gTLDs and the ccTLDs, the slave servers for those would provide immediate. A primary nameserver is the default for a name lookup. A query will failover to the secondary if the primary is not available.
  + 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.
    - 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
    - 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.
    - simultaneously floods harbor.ecn.purdue.edu with packets that look like reply from DNS server but contain wrong IP address. Each reply with a different Transaction ID, with hope that one of those fake replies match Transaction ID in query sent out by harbor.
    - there is now a race between correct reply from nameserver that has legitimate IP address for the amazon.com domain and the flood of fake replies sent by the attacker.
  + another weakness of DNS protocol 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 this attack. However, 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.
    - nameserver ns.purdue.edu have no entries for this hostnames and contact one of root nameservers for com domain and eventually contact the nameserver for the foo.com domain for the IP addresses for 1.foo.com, 2.foo.com, etc.
    - attacker sends spoofed replies from ns.foo.com to ns.purdue.edu for all of the queries from the latter for the various versions of foo.com hostnames. Attacker will have to race against the true answers sent to ns.purdue.edu from the authentic ns.foo.com.
    - a caching nameserver ns.purdue.edu would not only accept the Resource Records in fake replies to its queries, but also RRs in the Additional Section where attacker may even place a fake address for ns.foo.com and associate a long TTL with this entry.
    - Subsequently, any third-party accessing ns.purdue.edu nameserver for an IP address for any host in foo.com domain will reach the attacker nameserver instead of the true nameserver for the foo.com domain. Attacker could create any set of hostname-to-IP address mappings for the hosts in the foo.com domain.
  + fix for the problem above
    - Make it more difficult to take advantage of birthday paradox for guessing the Transaction ID in a query from a resolver or a recursive nameserver.
    - insisting that all recursive nameservers carry out bailiwick check of RRs in 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 outgoing query.
* Firewalls (both packet filtering and proxy server-based)

1. packet filtering: Linux kernel uses four tables, each consisting of rule chains, for processing incoming and outgoing packets. Each packet is subject to each of rules in a chain and the fate of the packet is decided by the first matching rule.:

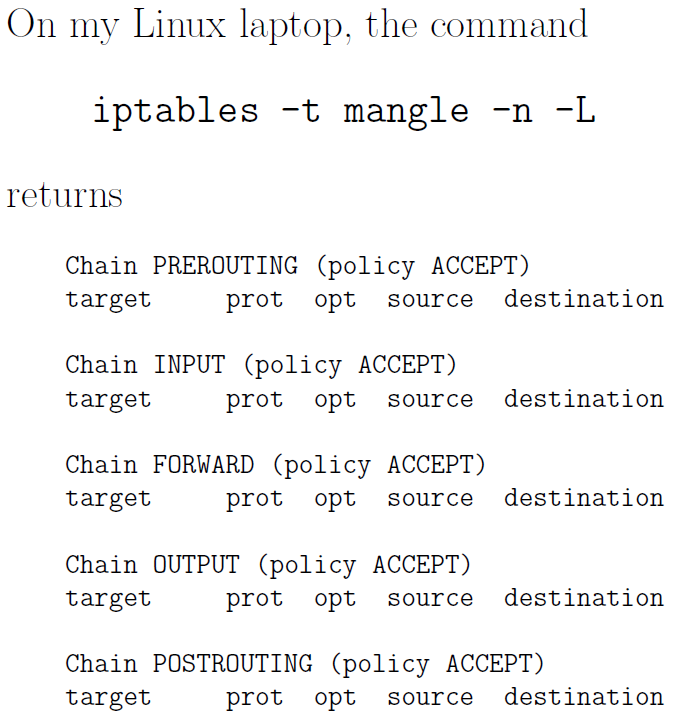
* + filter table: contains of three built-in rule chains that cannot be deleted)
    - INPUT: processing all incoming packets. When a packet comes in the kernel first looks at the destination of the packet. (labeled as routing). If packet is intended for the machine, packet passes to INPUT chain. If it is destined for another machine, packet goes to FORWARD chain.
    - OUTPUT: If a program running on computer wants to send a packet out of machine, packet must traverse through OUTPUT chain of rules.
    - FORWARD: processing all packets being routed through the machine. Only occurs when the machine is configured as a router.



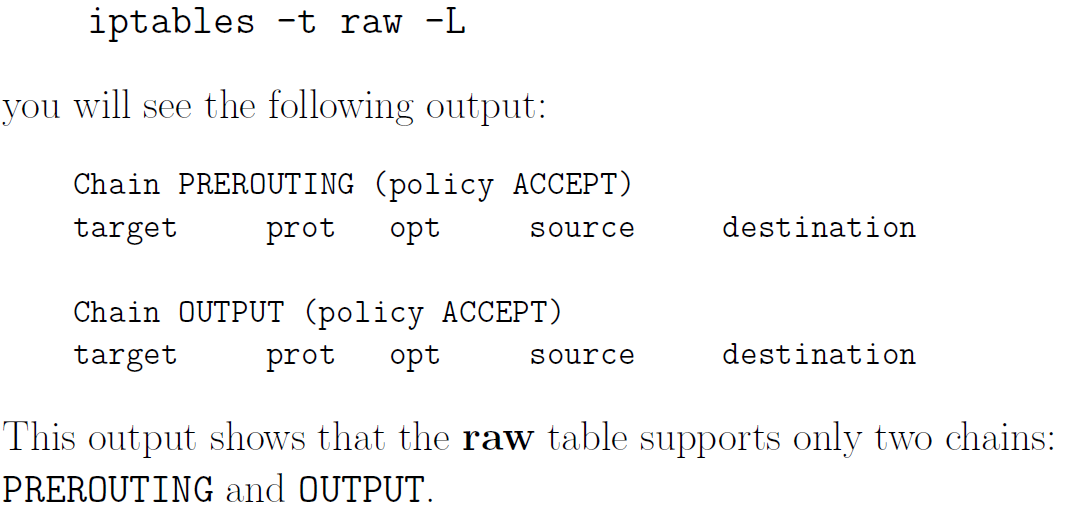
* If a packet reaches the end of a chain, Linux kernel looks at the chain policy to determine the fate of the packet. In a security-conscious system, this policy usually tells the kernel to DROP the packet.
  + nat table (Network Address Translation): When your machine acts as a router, it would need to alter either the source IP address or destination IP address, or both in the packet. Nat table consists of four built-in chains:
    - PREROUTING: altering packets as soon as they come in,
    - INPUT: altering incoming packets after they have been subject to pre-routing rules
    - OUTPUT: altering locally-generated packets before routing
    - POSTROUTING: altering packets as they are about to go out.
  + mangle table: used for specialized packet alteration, has five rule chains:
    - PREROUTING: altering incoming packets before routing decision is made
    - OUTPUT: altering locally generated outgoing packets,
    - INPUT: altering packets coming into the machine itself,
    - FORWARD: altering packets being routed through the machine
    - POSTROUTING: altering packets immediately after routing decision.
  + raw table: 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.
  + packet coming in for local host first be seen by mangle.PREROUTING, nat.PREROUTING, and mangle.INPUT chains before it is seen by the filter.INPUT. mangle.PREROUNTING and nat.PREROUTING would decide whether packet should be sent to a local process or forwarded to another communication interface. If the former is the case, the mangle.INPUT subsequently mark packet for downstream security based examination before the packet is subject to filter.INPUT.
  + STRUCTURE OF THE filter TABLE



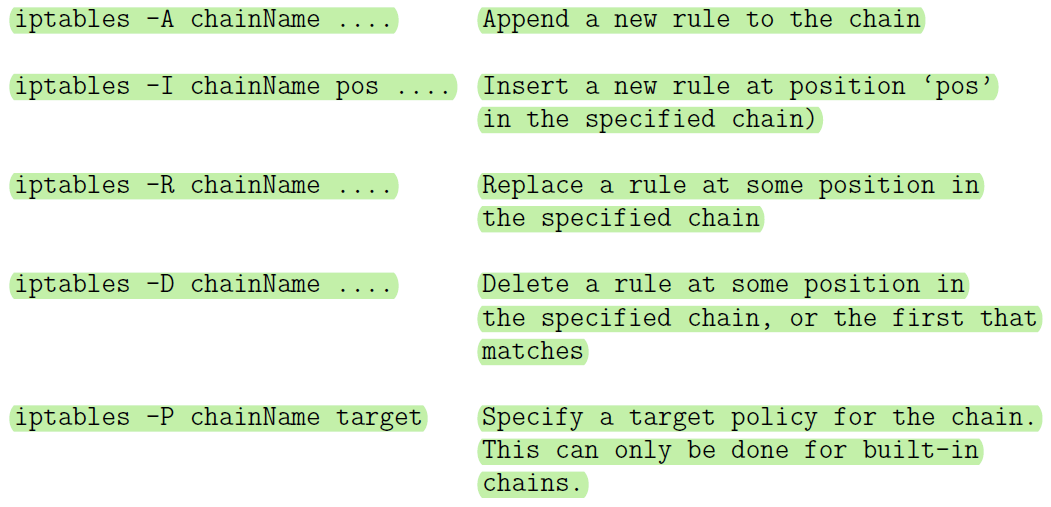
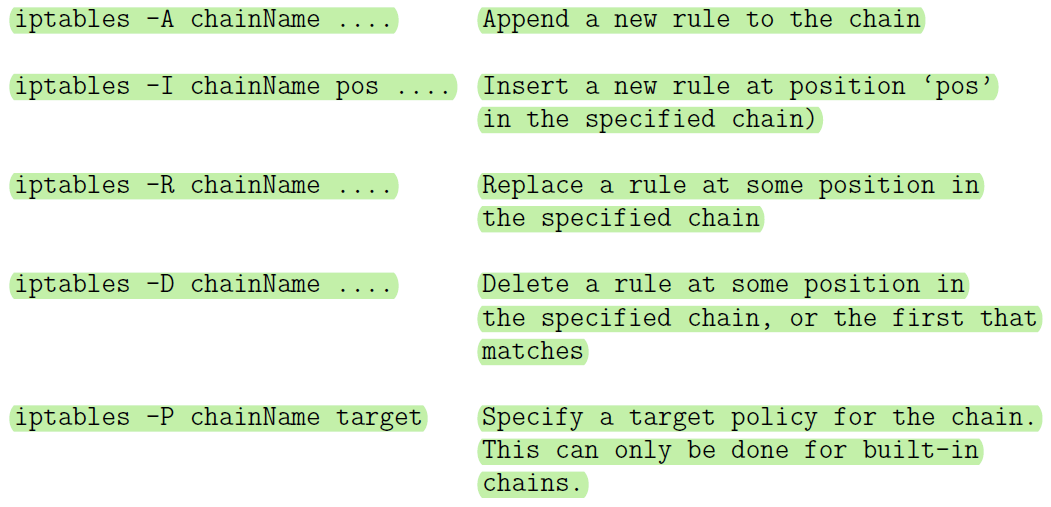
* + - num : The rule number in a chain.
    - pkts : The packet count processed by a rule so far.
    - bytes : The byte count processed by a rule so far.
    - target : action part of a rule, can be one of the following:
      * ACCEPT
      * DROP: drop the packet without sending an error message to the originator of that packet.
      * REJECT: drop the packet and sent an error message to the originator of that packet.
      * REDIRECT: send 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 : protocol associated with packet to be trapped by this rule. Protocol may be either named symbolically or a number.
      * when proto column mentions a user-defined service, then the last column must mention the port specifically.
      * for packets corresponding to standard services, system can figure out ports from the entries in the file /etc/services.
    - opt : optional
    - in : 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, you can request any service from your local system without the packets being denied.
    - out : 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, so all outbound packets will be sent directly 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 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. Targets for nat table:
    - 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 public IP address, your firewall 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 source address of local machine that houses server. But as these packets pass through the firewall, the source IP address in these packets is changed 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. Following targets can be used:

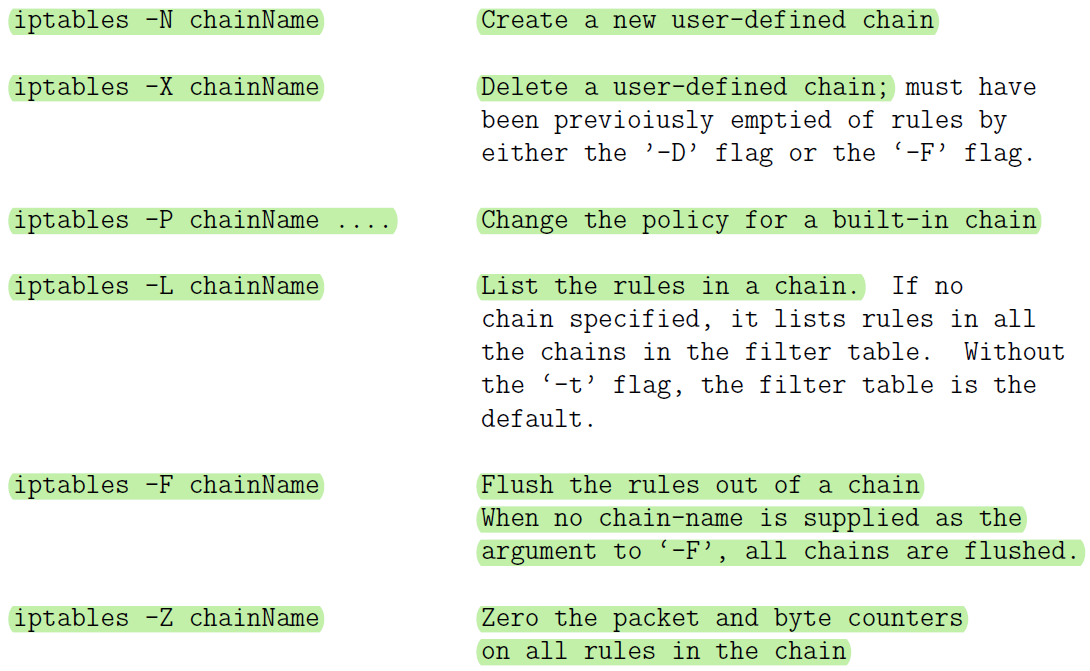


* + - TOS: change the TOS (type of service) field in a packet.
    - TTL: change the TTL field of the packet.
    - MARK: give a special mark value to the packet. Such marks are recognized by the iproute2 program for routing decisions.
    - SECMARK: sets up a security-related mark in packet. Such marks can be used by SELinux fine-grained security processing of packets.
    - 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 ! 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 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. 199.95.207.0/24 means all IP addresses between 199.95.207.0 and 199.95.207.255.
  + uses the net mask to specify a group of IP addresses. 199.95.207.0/24 is now accomplished by 199.95.207.0/255.255.255.0. If nothing comes after the forward slash, the default of /32 (/255.255.255.255) is assumed. Both imply that all 32 bits must match and only one IP address can be matched. /0 means every IP address.
  + ptables -A INPUT -s 0/0 -j DROP cause all incoming packets dropped.
  + 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.



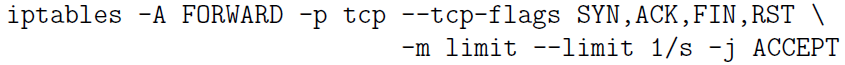
* how the interpretation of the results on a packet produced by the ip\_conntrack module should be carried out is specified by the additional ‘—state’ option.
* --state suboption supplies a comma-separated list of states of the packet that must be found to be true for the rule to apply. These states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



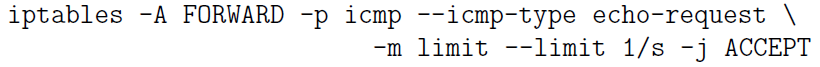
* + mac module: matching an incoming packets source Ethernet (MAC) address. This only works for packets traversing the PREROUTING and INPUT chains. It provides only one option --mac-source as in “iptables -A INPUT -m mac --mac-source 00:60:08:91:CC:B7 ACCEPT”or “iptables-A INPUT -m mac --mac-ource ! 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,

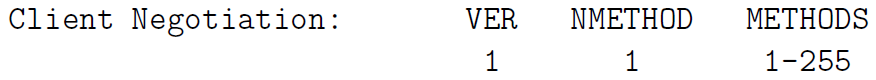


2. firewalls can be designed to operate at any of the following three layers in the TCP/IP protocol stack, often coexist for enhanced security:

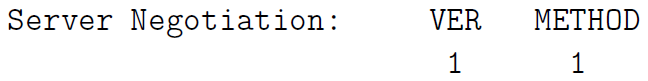
* + - Transport Layer: packet filtering. Examine every packet, check its IP headers and higher-level protocol headers to decide whether to let the packet through and whether or not to change any of header fields.
    - Application Layer: HTTP Proxy. examines the requested session for whether they should 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 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.

3. proxy server: 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

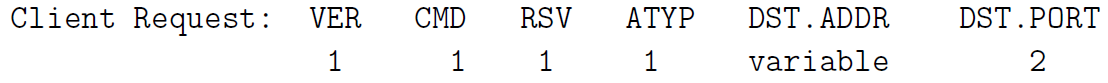
* SOCKS: a generic proxy protocol for TCP/IP based network applications, consists of two components: A SOCKS client and a SOCKS server.
* socks client is implemented between application layer and the transport layer
* socks server is implemented at the application layer.
* Socksifying the client call: 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 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 actual IP addresses of LAN hosts. With a SOCKS proxy, your IP address would remain hidden even it is a public address. Routing all network traffic through a proxy server makes it easy to centrally log all internet bound traffic and caching of web services.
* Interaction Between a SOCKS Client and a SOCKS Server: client wants to access an HTTP server in the internet
  + 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.



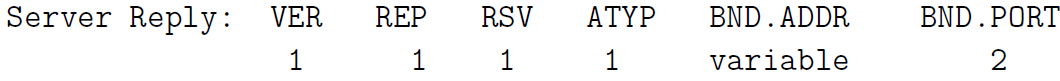
* + - * VER: the version number (SOCKS4 or SOCKS5)
      * NMETHOD: the number of methods
      * a listing of those methods by their ID numbers
    - server accepts the client packet and responds back with a Server Negotiation packet:



* + - * METHOD field: authentication method that server wishes to use.
      * socks server proceeds to authenticate the LAN client using the specified method.
  + socks client then sends 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.
  + 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)

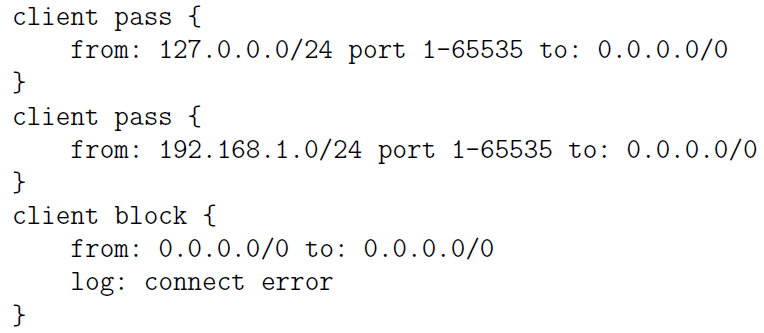


* BND.ADDR: is the internet-side IP address of the socks proxy server that the remote server will communicate with
* BND.PORT: is the port on the proxy server machine that the remote server sends the information to.
* REP: can take one of the following ten different values:
  + 0x00: successful connection 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
  + If connection between proxy server and remote server is successful, 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 nature of 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: IP address associated with the proxy server and the port it will monitor. What is needed is the IP address of the host (proxy clients) on which the proxy server is running.
        + external: IP address that all outgoing connections from server

IP address of the interface or name the interface (such as eth0) on which the proxy server will be communicating with rest of the internet

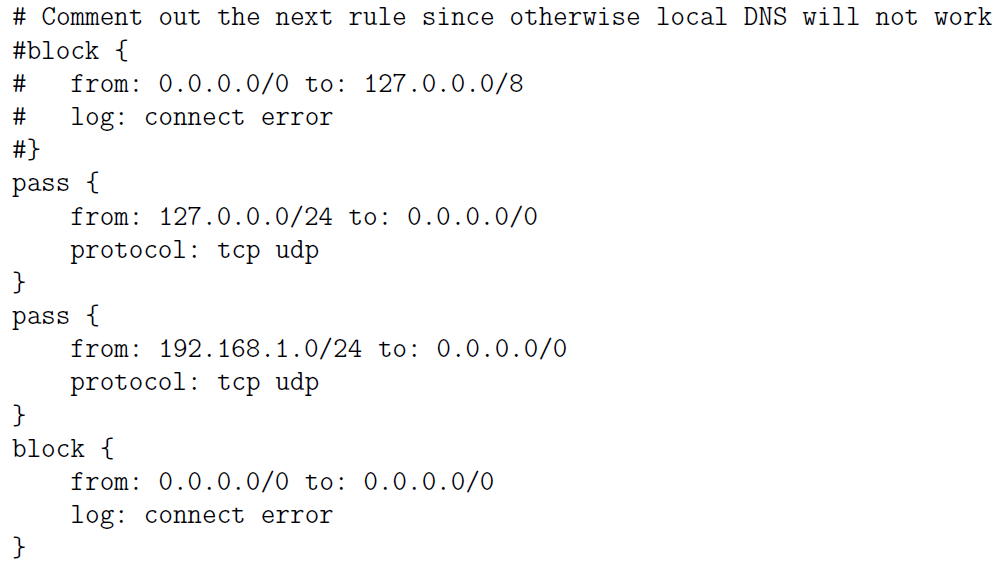
use a socks proxy behind a home or a small-business router wont gain any anonymity from the outgoing IP address used by the SOCKS server since the router will translate the outgoing IP address into what is assigned to router.

* + - * + method: 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.
        + user.notprivileged: what read/write/execute privileges the server should be set to when running in the default non-privileged mode.
      * Rules
        + client rules: which socks clients are allowed to talk to the proxy server.



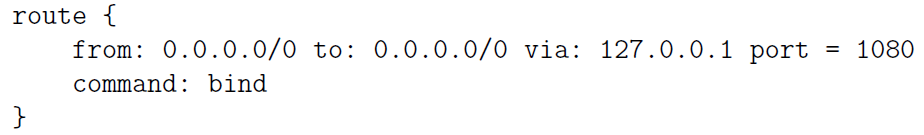
These rules allow all local socks clients on the same machine and all socks clients on local LAN to talk to the SOCK proxy server. The third rule deny access to all other socks clients.

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

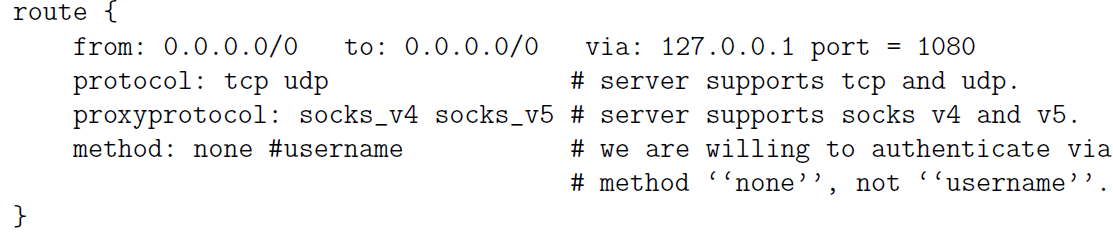


any local socks client will be able to call on any service anywhere for a TCP or UDP service. The third rule does the same for any socks client in the local LAN. The fourth rule blocks all other socks client requested services.

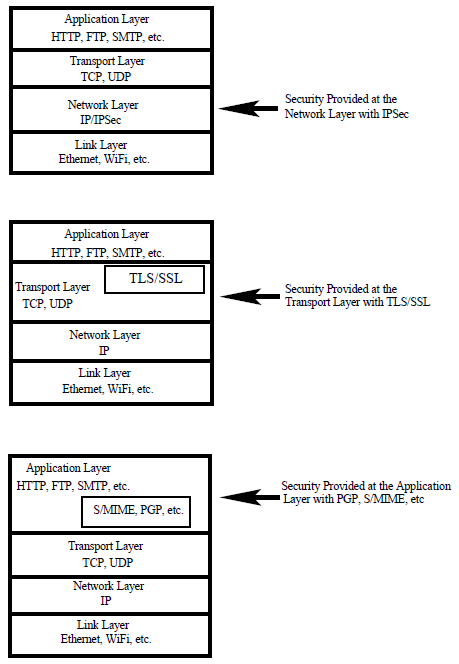
* + - * Routes
    - client configuration file /etc/dante.conf: regulates the behavior of a socksified client
      * bind command allows incoming connections for protocols like FTP in which the local client first makes a control connection with a remote server and the remote server then makes a connection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

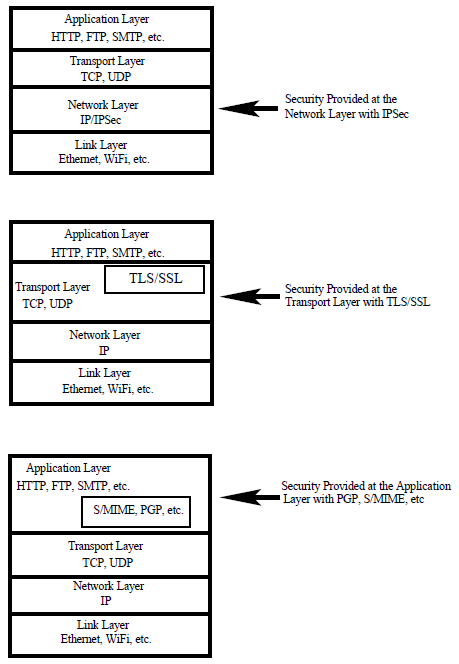


* The following rule tells client where SOCKS proxy server is located and what port the server will be monitoring, the server supports TCP and UDP, SOCKS4 and SOCKS5 protocols, and the server does not need any authentication

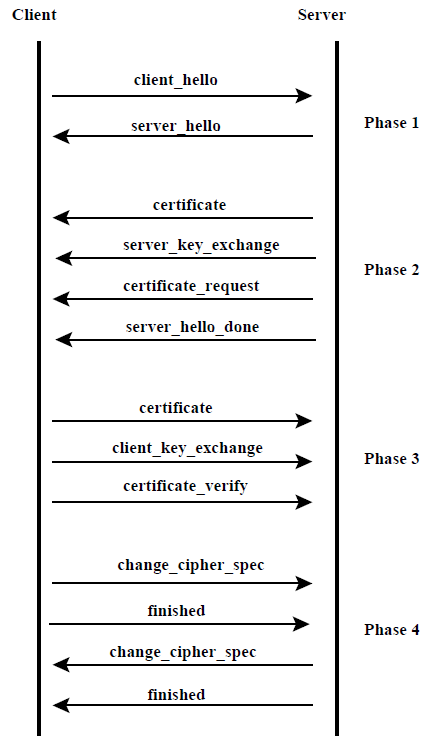


* + Squid: can be used as an anonymizing proxy server to control access to web (HTTP, HTTPS) and FTP resources
* Usage
  + Through combination of ACL (Access Control List) and http access directives in config file to control who is allowed to connect with proxy from what IP.
  + To designate a list of 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 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, proxy server will send out cached page.
  + parent-child: If a child cache cannot find an object, it passes on request to parent cache. If the parent cache does not have the object, it fetches and caches object and passes it to on to the child cache that made the original request.
  + Sibling: 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 automatically expires clients objects to avoid returning to the objects that are out of date. You can set the refresh time in the configuration file.
* Squid has the notion of Safe ports, 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 not, they would not go through the server.
* 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 established with SSL/TLS protocol are certificates issued by Certificate Authorities (CA).
    - server-only authentication:
      * client receives and verifies server’s certificate, generates a secret key that it then encrypts with the server’s public key.
      * client sends the encrypted secret key to the server; the server decrypts it with its own private key and use it to encrypt the messages meant for the client.
    - server-client authentication: in addition to secret key, the client also sends to the server its certificate
* SSL is composed of four protocols in two layers
  1. SSL Handshake Protocol: authenticates clients and servers to each other
     + Generate information about algorithms for compression, authentication, and encryption in SSL Record Protocol.
     + Generate cryptographic keys for the encryption and the authentication of each SSL record.
     + SSL Handshake protocol works in four phases.



* + - 1. Phase1: initiated by client, to establish the security capabilities present at the two ends of a connection. The client sends to server a client hello message:

Version (highest SSL version understood by client)

Random (timestamp and random field together serve as nonces during key exchange to prevent replay attacks)

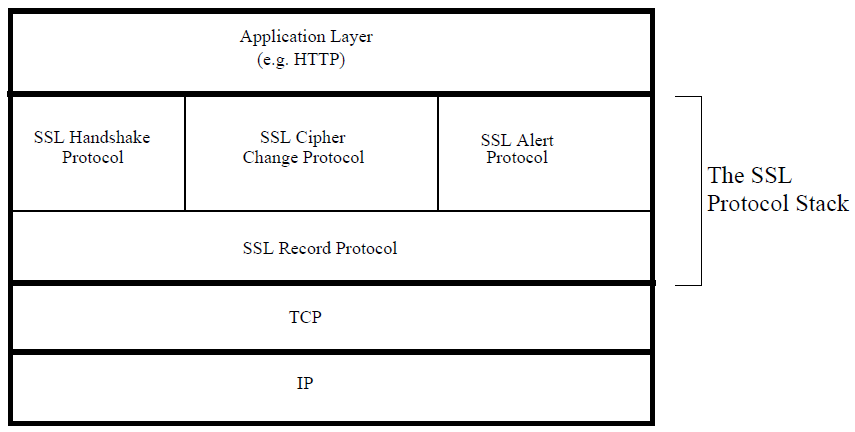
Session ID (a variable length session identifier)

Cipher Suite (a list of cryptographic algorithms supported by the client);

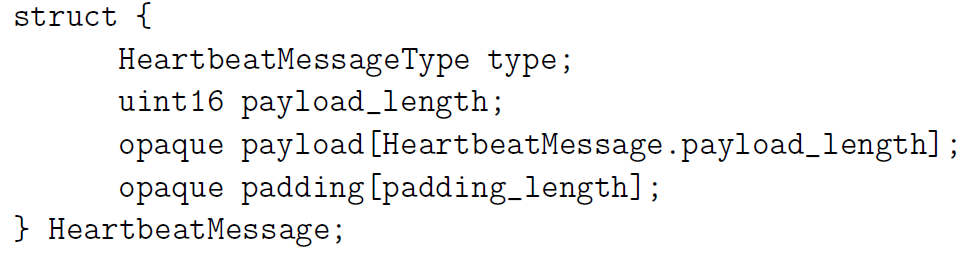
Compression Method (a list of compression methods the client supports)

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.

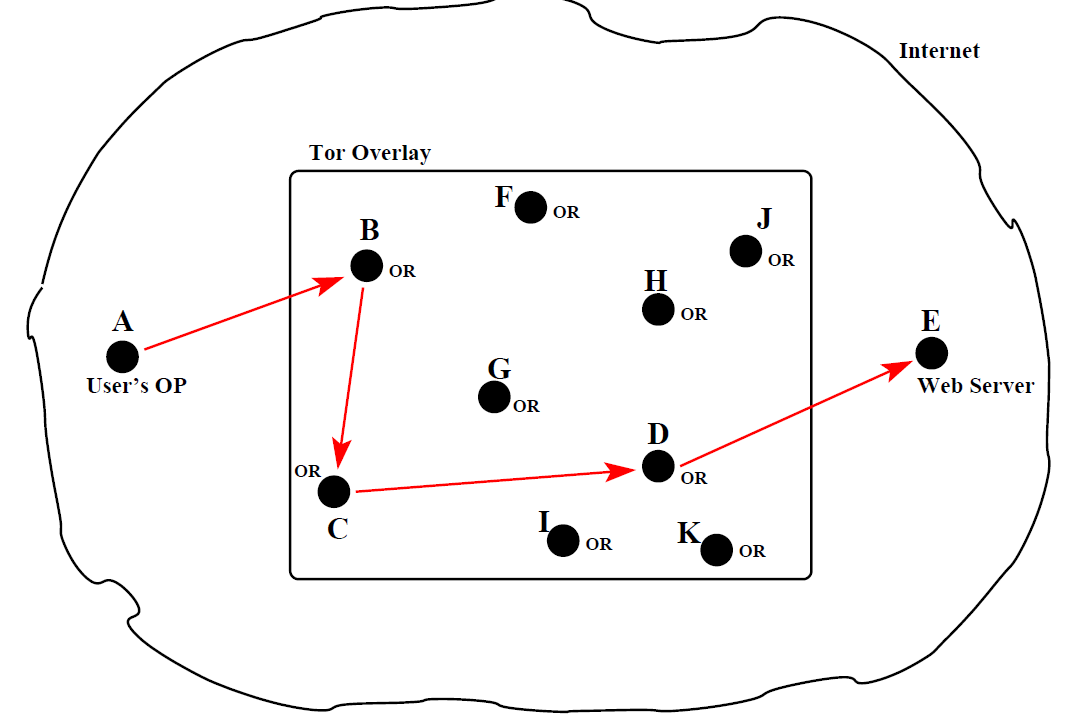
1. Key exchange method: RSA, Diffie-Hellman, etc.
2. CipherSpec: indicate the authentication algorithm selected, length of MAC, encryption algorithm, etc.
   * + 1. Phase2: server sends client a message labeled certificate containing its certificates for the validation of the server public key. This could be followed by a server key exchange message, and a certificate request message. Phase 2 handshaking ends when the server sends the client a server hello done message.
       2. Phase3: client sends server the client certificate if server made a request. Next, client sends server a client key exchange message contains a secret session key encrypted with server’s public key. This phase ends when client sends server a certificate verify message to provide a verification of its certificates.
       3. Phase4: completes the setting up of a secure connection between client and server. The client sends server a change cipher spec message indicating that it is copying the pending CipherSpec into the current CipherSpec. Next, the client sends 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
      * 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: 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: convey SSL-related alerts to the peer entity.



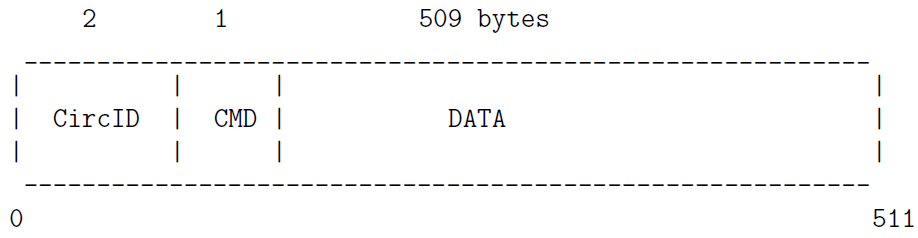
* + Connection vs session: connection takes care of transferring information securely from one endpoint to the other, session allows for such data transfers to take place back and forth without having to renegotiate the security parameters for each separate connection.
  + SSL Connection: a one-time transport of information between two nodes in a communication network. Every connection is associated with a session. SSL connection state is characterized by the 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, 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: an enduring association between a client and a server.
    - A session is created by the SSL Handshaking Protocol.
    - A session can consist of multiple connections.
    - A session is characterized by a set of security parameters that apply to all the connections in the session.
    - A session comes to an end when the exchange of data between the two endpoints has come to an end.
    - 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 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
* Central to 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 lifetime of 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.
* 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: 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.



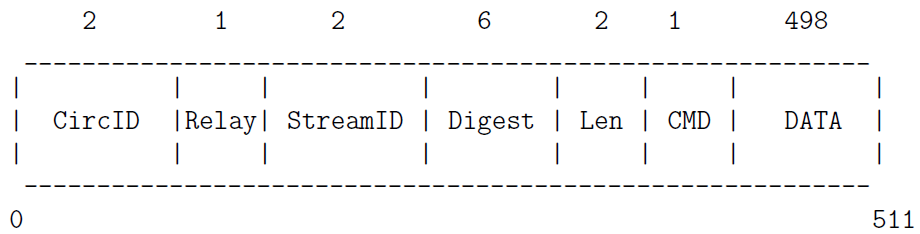
* Tor (The Onion Router) protocol for anonymized routing: a way to set up internet communications that an adversary would not be able to analyze packet headers for the purpose of finding out who was talking to whom.
* Tor protocol interplay between RSA public-key cryptography and the DH (Diffie-Hellman) public-key cryptography.
* Tor protocol is based on 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 3, for constructing a path to the destination.
* 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.



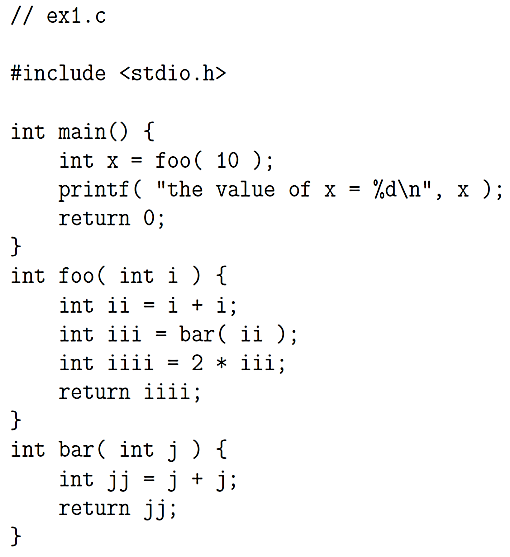
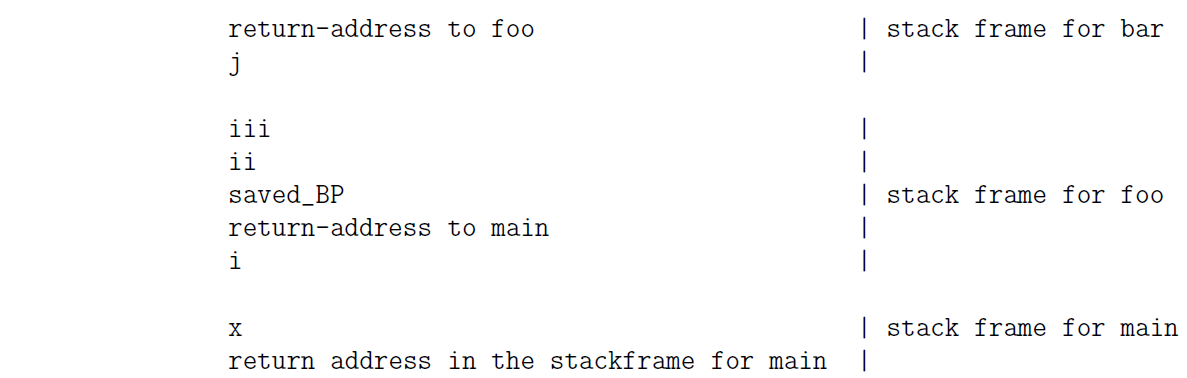
* a message that is exchanged between an OP and an OR or between two ORs is called a cell / torpacket. Each torpacket consists of 512 bytes, control and relay torpackets work together to create an end-to-end path in Tor overlay that each interior node on path has only local knowledge of path.:
  + control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



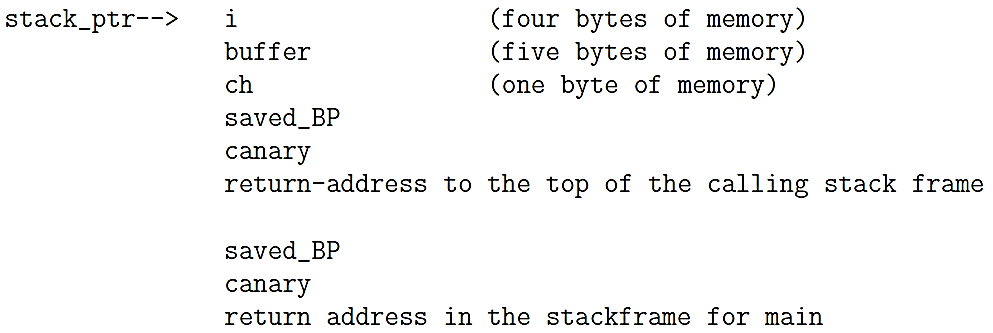
* circID : a circuit identifier is unique to each hop in a circuit despite the fact that the circuit abstraction applies to entire end-to-end path.
* CMD: control torpacket may contain the following commands:
  + create : sent by an OP or OR to another OR to extend the path to the next node
  + created : 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):
  + 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 an OP uses 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 as 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 created control torpacket: DATA field of this torpacket contains Bs DH Y key Y\_BA. Now both A and B can calculate secret session key K\_AB for their link
      3. Nodes A and B start exchanging relay torpackets, using identifier circID\_AB for circID field: A sends B a relay torpacket with relay extend command to extend the circult: DATA field is a DH Y key Y\_AC encrypted with C’s RSA public key, so cannot be seen by node B. DATA field in relay extend torpacket from A to B is encrypted with session key KAB.
      4. When B receives the relay extend torpacket from A, it generates a control torpacket whose DATA field contains Y\_AC encrypted with C’s RSA public key. The control torpacket sent by B to C uses a new randomly generated number as circID\_BC. Only node B knows both circIDAB and circIDBC so 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. 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. path may be extended in the same manner to the node D
      7. there was never a need for using As public RSA key. User A remains anonymous to all 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. 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, user A start pushing data into the circuit that is meant for the final destination E.
      10. stream data that the user A places on the wire is encrypted with 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 node. 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.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. Exit node operator obviously cannot peer inside the packets that A is sending out.
* 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 are not stored in heaps, it is more difficult to launch functions exploits with heap overflows than with stack overflows.
* 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
  + Stack Pointer: 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.

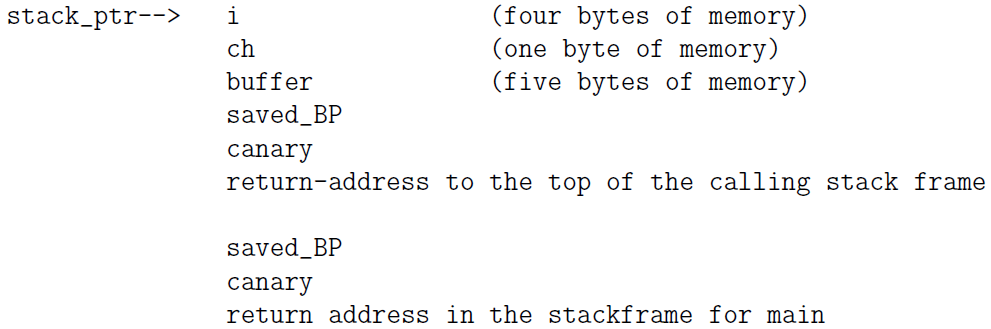


* Buffer Overflow Attack: Overrunning the Memory Allocated on the Call Stack. EX. the string you enter will begin to overwrite the memory locations allocated to other variables on the stack and also possibly the location where the return address of the calling function is stored. When this happens, the 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.
* 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. 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

* 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: return address in a stackframe is replaced by 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
* Viruses and Worms

1. 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). To make it more difficult to be detected, mutating viruses alter itself when it propagates from host to host.
   * Viruses typically place a signature (such as an impossible date) at a specific location in 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.

2.. 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.

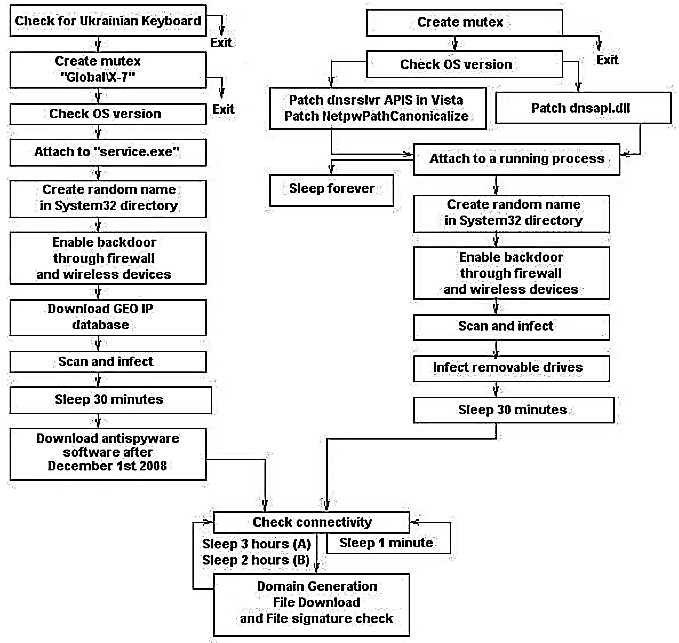
* + A program may hop from one machine to another by means that include:
    - using the remote shell facilities provided by ssh, rsh, etc., to execute a command on the remote machine and install a small bootstrap program on the target machine
    - crack passwords and log in as a regular user on a remote machine
    - 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. If propagation is fast enough, it can shutdown machines on network. This can happen particularly when 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 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 try to execute the named file. 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 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 rest of program execution to be not as originally intended. If an executing program allocates memory for a buffer on stack, but doesn’t 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 with 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 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, svchost process checks services part of registry to construct a list of services (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 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. 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
        1. download a copy of the worm using the HTTP protocol from another previously infected machine and store it as a DLL file
        2. execute a command to get a new instance of the svchost process to host the worm DLL
        3. enter appropriate entries in registry so that worm DLL was executed when the machine was rebooted
        4. gave a randomly constructed name to the worm file
        5. continue the propagation:

Windows function NetpwPathCanonicalize() exported by netapi32.dll over an SMB session on TCP port 445

in 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 use redirected return address to invoke URLDownloadToFile() to pull in the worm file.

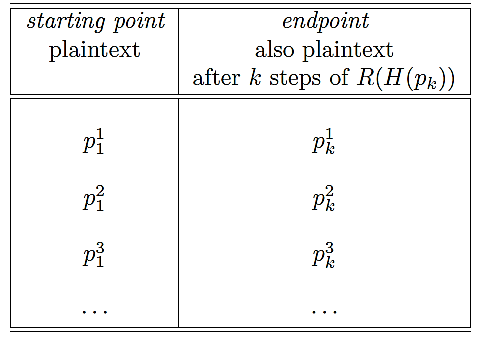
Once worm file is pulled into machine, it could be launched in a process/thread as a new instance of svchost.exe by calling LoadLibrary() whose sole argument was name of newly downloaded worm file. The LoadLibrary command also copied the worm file into the system root.

* + - * 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 can reset the system restore points, so an infected machine could not 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 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.
      * Patch dnsapi.dll: it compromises DNS lookup in machine to prevent the name lookup for organizations that provide anti-virus products.
      * 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
      * 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 random number generator used for this is seeded with current date and time, we can expect all 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.
  + STUXNET WORM: designed to attack industrial software SCADA to harm processes related to 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 frequency converters used to control centrifuge speeds to raise their frequencies to a level that would cause 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:
      * 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
        1. DoublePulsar: exfiltrate documents from victim machine and pull in additional malicious code, such as WannaCry, from network. Remains installed in victim host even WannaCry is removed. Resides only in the RAM of a host. It is memory resident malware and disappears when you reboot host.
        2. 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).
        3. 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.
        4. 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: use a large number of attributes to characterize 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.
* Password Protected Systems
  + dictionary attack: commonly used to break into port 22, try a large number of commonly used account names on the target machine and if there is an account match, then try a large number of commonly used passwords for that account.
  + log scanning: protect a computer/network against a dictionary attack
    - With log scanning 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,
    - there is a fundamental difference in how the two tools keep the blacklisted IP addresses: (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
* Fail2Ban: detect intrusion attempts by keeping track number of login attempts. Using regex based filters, Fail2Ban detects malicious behaviors by connections made by IP addresses and take action you wish to those IP addresses.
  + Fail2Ban’s 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 identify IP addresses that are engaged in different malicious activities and, depending on what activity is involved, it can take different actions.
  + Parameters in /etc/fail2ban/jail.conf
    - Bantime: 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: mail transport agent to use for sending 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. With the action\_mw, email notification will include a whois report on the intruding host. With action\_mwl, email notification will include relevant log lines.
  + DenyHosts
    - in addition to entering a blacklisted IP address in /etc/hosts.deny file, the blacklisted IP addresses are also recorded in a few more files in your directory system for synchronizing your blacklisted IP addresses with similar such addresses collected by other hosts in the internet if you have synchronization option turned on in config files
    - main config file for DenyHosts is /etc/denyhosts.conf.
      * DENY\_THRESHOLD\_INVALID: limit on how many times an intruder can try to entry with usernames that don’t exist in /etc/passwd file
      * DENY\_THRESHOLD\_VALID: limit on trying to gain entry through usernames that do exist.
    - DenyHosts makes its log entries in the file /var/log/denyhosts
  + Password cracking: you broken into a machine and got System Password File where all password hashes are stored. Now you want to map password hashes back to the character strings that are the passwords as entered by the users.
    - 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 samples the space of all possible passwords. 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



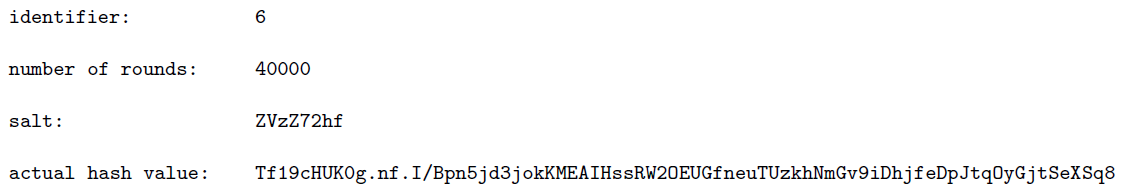
If any of q1, q2, … match any of the endpoints in the second column of the table, then there is a high probability that the password is in the chain corresponding to that row.

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

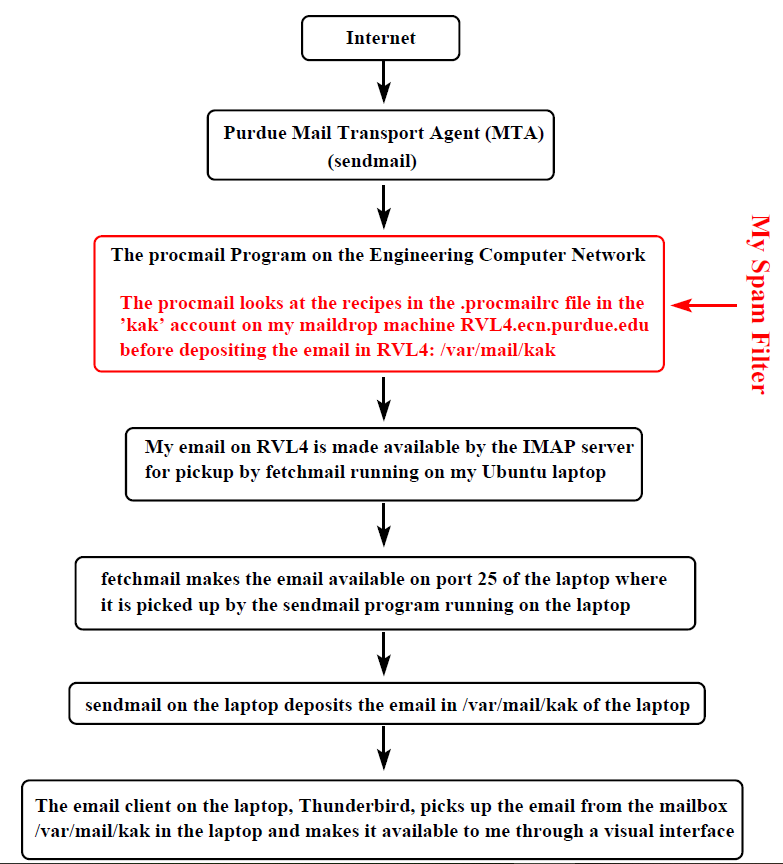


With a significant probability, the cracker will find that his hash C matches one of the hashes in this chain. Once a match is found, the password is the plaintext that immediately precedes C in the chain.

* + - * as we grow test hash chain one step at a time starting with hash C, we run into a qm that matches one of endpoints in table, but we are unable to find C in the chain for that row. In this case, we continue to grow test chain and look for 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 chains span 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 starting points and endpoints for chains. Collision 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 upper bound 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. As the size of the table grows, the table becomes more and more inefficient on account of chain merging. Before 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.
      * in hash-chain tables, you may encounter a loop as you grow a chain. Since a reduction function is intentionally many-to-one, there is a chance that password that is reduced to will be the same at two different places in a chain. However, when you use different reduction functions for successive reduction steps in a chain, you are less likely to run into loops.
      * lookup consists of first applying the last reduction functions R\_k() to obtain q1= R\_k(C) and checking whether q1 is an endpoint in rainbow table. If not, we grow test chain by calculating q2 = R\_k-1(H(q1)) and search for q2 as an endpoint in the table and so on.
  + hash-table database: construct a hash for all possible character combinations and store these <password, hash> values
  + lookup-table attack: password hash is attacked by looking up a table of previously computed hashes
  + two facts of the development of password cracking method
  + ubiquity of the Windows machines all around the world.
  + older versions of Microsoft Windows platform used 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 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 original password string was shorter than 8 characters since the second half input string is all zeros and it results in the predictable DES encryption given by the hex 0xAAD3B435B51404EE.
      * 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 69^7 + 69^6 + 69^5 + 69^4 + 69^3 + 69^2 + 69.
  + reason why you cannot just directly apply an algorithm such as SHA-512 to a user-entered password string: 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 the user-chosen password string with salt (a number of random bits)
  + 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



* Email/Spam Filters
  + Bayesian filters: statistical filter with sufficiently low falses 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 and all its variations that the spammer may use to block the email
  + MTA (Mail Transfer Agent) / Mail Transport Agent / Mail Exchange Server: 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.
  + email sending process
    - client email goes to an MSA (Mail Submission Agent)
    - MSA forwards email to MTA.
    - MTA forwards email to MDA (Mail Delivery Agent). MTA can also be programmed to send email directly to the clients.
    - MDA send email to clients.



* + Send an email to [kak@purdue.edu](mailto:kak@purdue.edu)
    - name resolver associated with email client used by sender ask DNS servers for IP address of the host that is designated to be the mail exchange server for the purdue.edu domain.
    - MTA program running on this host at Purdue receive the email
    - MTA may use either a Mail Delivery Agent (MDA) to deliver email to recipients mailbox, or deliver it directly to recipients mailbox.
    - MDAs main job is to apply filters to email before it is deposited in the mailboxes of user accounts. These filters may be at the system level to affect all users, or at the level of individual users. Filters used by MDA take the form of recipes placed in a file .procmailrc.

.procmailrc file for individual users must reside at the top level of the user-account home directories.

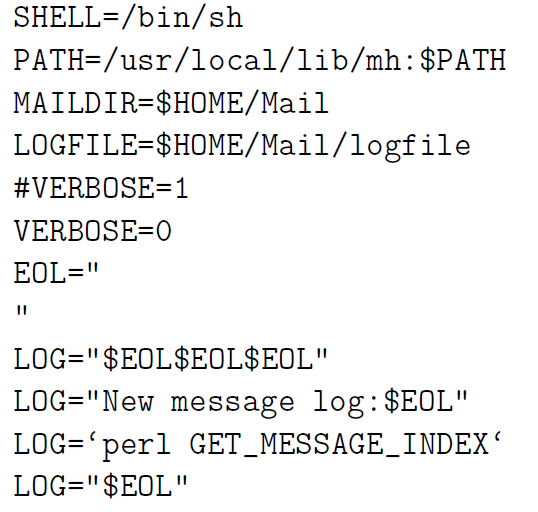
* + - After email is deposited in a user mailbox, it may be read by user with help of an MUA (Mail User Agent), also called email client.
    - read my email:

ssh kak@rvl4.ecn.purdue.edu

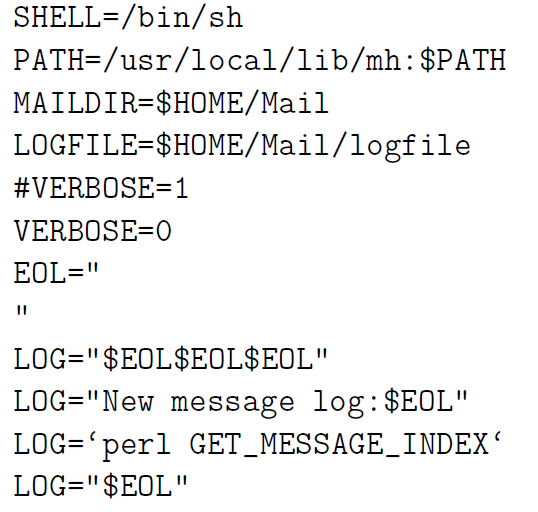
tail -f Mail/logfile

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.

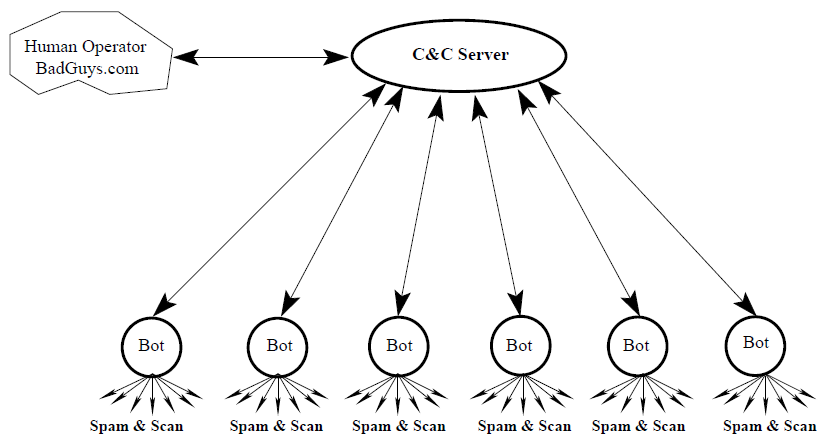
* + An email consists of three parts
    - header: Contains From:, To:, Cc:, etc. The header of an email ends at the first empty line. What comes after is the body of the email.
    - body: carries message of email, may contain multimedia objects.
    - envelope: This part is usually suppressed by an MUA. 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 MTA, its first line must begin with From. 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 HTML.
  + Recipient’s name in the envelope of an email that determines where an email ends up. NOT what shows up in To: in the header part of an email.
  + MUA may modify the first From line into two separate lines, one for Return-Path and the other for Delivery-Date, so what MTA sends MDA may not be the same as what the MUA stores for the email and shows you on the screen.
  + .procmailrc file consists of three parts:
    - Assignment of relevant environment information to local variables



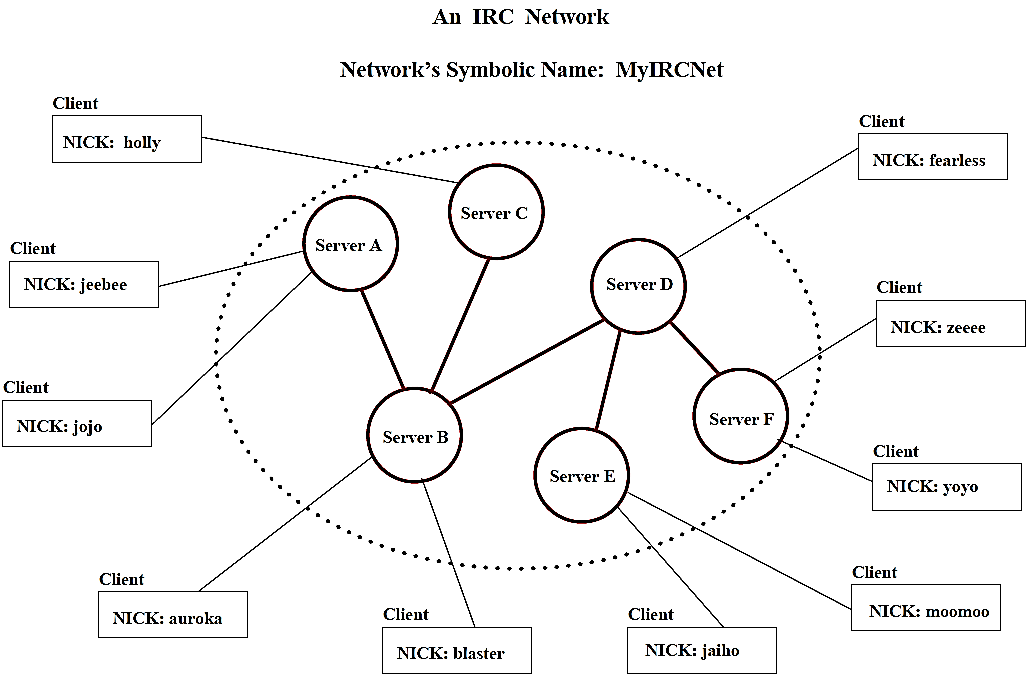
* + - Assignments to variables that will be used as macros in .procmailrc



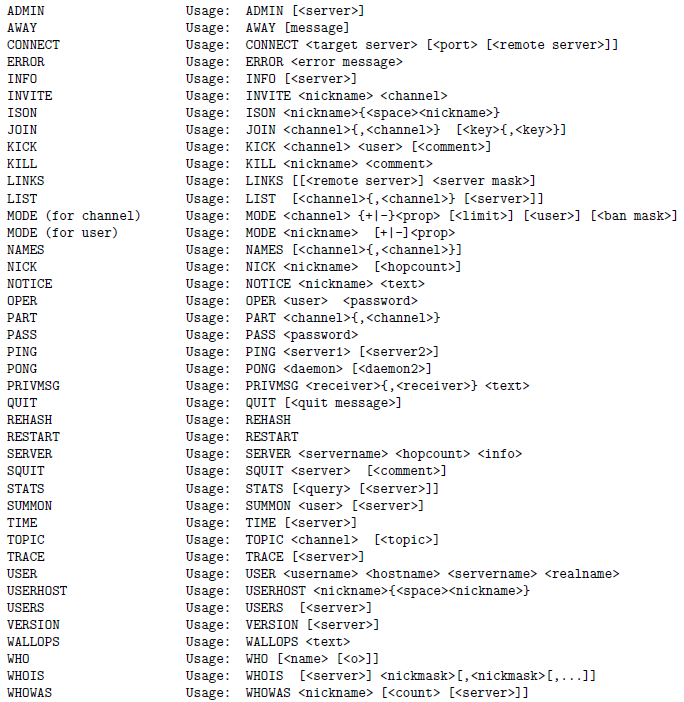
* + - Recipes
      * colon line: “:0 [flags] [ : [locallockfile] ]”. D: case-sensitive.
      * condition line: always begins with “\*”. There can be only one condition per line. Any white space between \* marks start of a condition-line and the first non-blank character that comes after that in the same line is ignored. !: Invert condition. <: Check total length of email is less than number of bytes that is specified after this character. “\* ? $MAILDIR/condfilter2.pl 2>&1”: recipe feeds the email into the Perl script condfilter2.pl. The condition succeeds if Perl script returns 0 and fails if it returns 1.
      * Action line: !: email is forwarded to email address comes after this symbol
      * Delivering Recipes: cause email to be written to a file, or forwarded to another email address, or absorbed by a program. Procmail quits processing the email.
      * Non-delivering Recipes: cause output of a program to be captured back by Procmail. procmail continues processing this output in the same way it processes as a regular email.
      * delivering recipe can be made to behave like non-delivering recipe by specifying c flag in colon line. c flag causes a copy of the email to be sent to delivering recipe while the original is saved for processing by the rest of the .procmailrc file.
* Bots
  + viruses and worms are equipped with a certain fixed behavior. Bot is equipped with a larger repertoire of behaviors. Additionally, a bot maintains link with a human handler (bot-master or a bot-herder.)
  + A bot master can harness the power of several bots working together. For example, mount a distributed denial of service (DDoS) attack
  + Botnet: A collection of bots working together for the same bot-master
  + two different ways for a bot may receive commands from its master, Both of these modes require a command-and-control (C&C) server that talks to the individual bots:
    - push mode: C&C Server broadcast the same message to all the bots, sends or pushes the command and control messages into the bots. (IRC protocol)
    - pull mode: bots send a request to the C&C server every once in a while for the latest commands (HTTPD protocol)



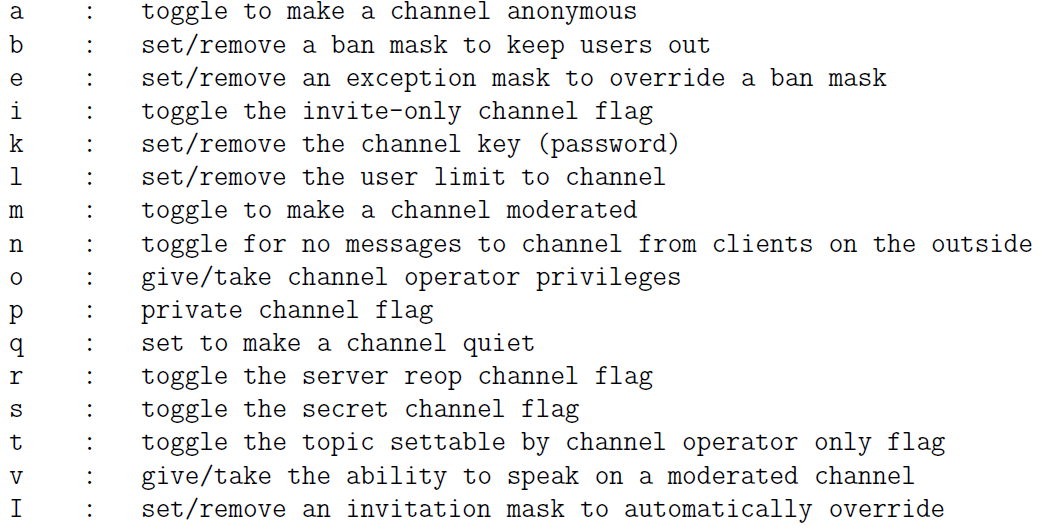
* + a botnet exploit is more likely to go undetected if the communication between the bots and C&C server uses standard protocols. Botmasters of today are increasingly favoring pull mode over push mode. Since HTTP traffic is more pervasive in the internet compared to IRC traffic
  + each bot registers itself with C&C server, so bot master only has to communicate his/he intentions to C&C server for those intentions to be sent to all bots. Therefore, communications between human and C&C server is infrequent, making human handler hard to discover.



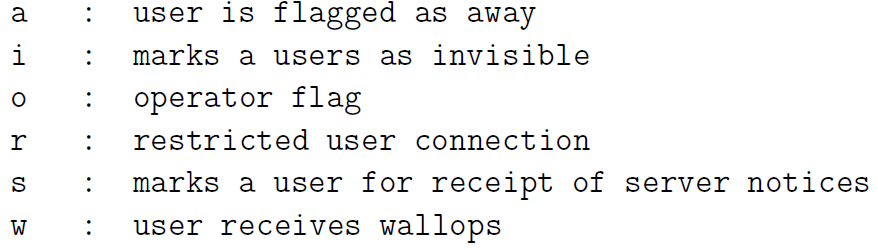
* + IRC (Internet Relay Chat):
    - individual chat clients could be plugged into different machines in different parts of world, these different machines in the same IRC network would appear as a single logical chat server to all clients. Therefore, all of individual servers must stay synchronized and IRC protocol cannot easily be scaled up to an arbitrarily large number of servers
    - An IRC overlay is not allowed to have loops (spanning tree over the underlying TCP/IP network), so there is always a unique path from any one server to any other server and each server node can act as a central node vis-a-vis the rest of the IRC network. It makes it easier to update all the servers in real time with the latest information regarding the servers and the users. It is the responsibility of each server to forward all the received state information to the servers it is connected to
    - Nick: Each user in an IRC network is identified by a nickname
    - Channel: a set of users. There are two kinds of channels
      * local to each specific server
      * global to all the servers.
    - When a message is sent to a channel, it is sent to all the users that are in the set corresponding to the channel.
    - IRC protocol considers the first person to start a new channel as the operator of that channel. After you have identified yourself with the IDENTIFY command to ChanServ, you will always have your operator privileges
    - messages syntax in an IRC network
      * an optional :-prefixed string, followed by
      * a valid IRC command in ASCII (or the corresponding 3-digit number), followed by
      * the arguments to the command.
      * when you as a client send a message to the server: MODE #botnetUnderground +k abracadebra
      * when the same message is forwarded by the server that received your message to other servers: :botBoss MODE #botnetUnderground +k abracadebra
    - IRC protocol is a line-oriented protocol: An IRC message is always terminated in the internet line terminator CR+LF.
    - use IRC for botnets, channels can be made secret and users made invisible by setting properties.
    - commands of the IRC protocol:



* + - Set property for channel: MODE <channel> {+|-}<prop> [<limit>] [<user>] [<ban mask>]
      * EX. “MODE #botnetUnderground +s” make channel #botnetUnderground secret.
      * EX. “MODE #botnetUnderground +k abracadebra” sets key for the channel



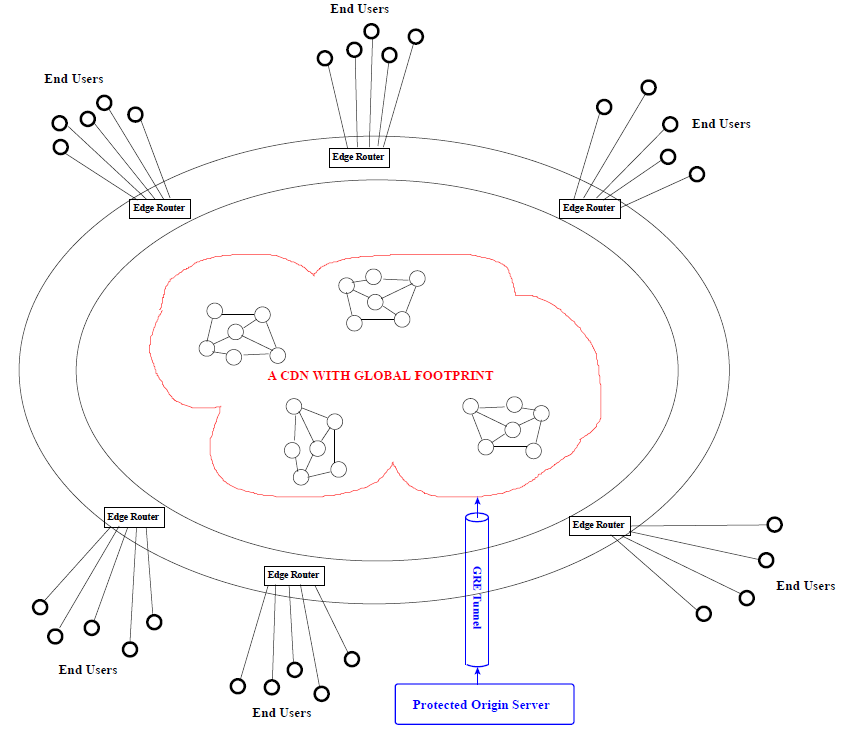
* + - Set property for user: MODE <nickname> [+|-]<prop>
      * EX. “MODE botBoss +i” marks user botBoss as invisible.



* + - command for sending text to other users: “PRIVMSG #botnetUnderground :Hello Bots! Are you ready to wage war?” The message Hello Bots! Are you ready to wage war? will be sent to all the users who are members of the channel #botnetUnderground
* DDoS attacks: to overload a network with massive amounts of contrived traffic so that it becomes unusable by its legitimate users.
  + Three types of DDoS attacks
    - Volumetric DDoS Attack: Bandwidth exhaustion is a form of.

Volumetric DDoS Attack. It causes maximum possible exhaustion of network resources at a targeted host.

* + - TCP State Exhaustion Attack: any computation related to the operation of the TCP/IP engine can only support a certain maximum number of processes (or threads) running concurrently. The goal of this attack is to commandeer all available concurrency at the targeted host.
    - Application Layer Attack / Layer 7 DDoS Attacks: flood an application at a targeted host with routine looking requests. HTTP GET and POST floods are examples of such attacks. Since such attacks can be mounted with a small number (even just one) of attacking hosts and since the traffic generated by such attacks looks like normal traffic, this type of a DDoS attack can be difficult to detect.
  + DDoS attacks based on the NTP and DNS amplification exploits(A is attacker (bots in a botnet), S is a DNS server, T is the victim):
    - A sends a large sequence of packets (with spoofed IP address of T as source) to S for a name lookup request.
    - Server S sends its response back to T
    - When the size of the response from S is k times the size of the request received by S. The attacker A can take advantage of this fact to create a large bandwidth burden for T without having to bear the same bandwidth cost himself.
  + DDoS attacks based on TCP: hit the TCP with a pulsating flood of DDoS packets every RTO seconds so that the sender TCP will never receive an ACK within RTT. Such attacks can easily go unnoticed even as the users of the internet are seeing a significant performance degradation in data download speeds from the internet.
  + methods to protect networks against DDoS attacks
    - multi-layer switching:
      * multi-layer switch can route packet base on information corresponding to any of layers 3 and above.
      * all functions implemented in hardware so that packet forwarding takes place at wire speed.
      * Layer 3 switch is no different from a router, Layer 4 switch carries out port translation for sending incoming packets to 1+ machines hidden behind a single IP.
      * Content switches: Layers 4-7 switches, used for load balancing when services are provided through a Content Delivery Network (CDN), a client can be connected to the least loaded node of a CDN.
      * If there were to be a DDoS attack a network of servers behind a multi-layer switch in a high-bandwidth local network. Switch can mitigate the attack by sending the incoming traffic to the least loaded server machine.
    - packet filtering at the routers;
    - providing services through Content Delivery Networks.
      * CDN: network of geographically distributed customer-facing proxy servers that deliver the content in the internet. The origin server cannot be reached directly by the internet users. Isolating the origin servers makes them relatively secure against DDoS attacks
      * origin server supply content to CDN proxy servers through GRE tunnels, a secure point-to-point tunnel created with Generic Routing Encapsulation Protocol
      * An attacker could try to defeat a CDN by making rapid requests for dynamically generated content that would cause CDN to go to origin server for fetching content. But in practice it can be warded off by aggressive caching both at CDN end and at the origin server end.
      * Even with aggressive caching, there are certain types of requests that the CDN would need to send back to origin server: login requests, search, etc. So DDoS attack that exploits such requests may succeed. The origin server can be protected against such attacks with appropriate content switching and rate-limiting firewall rules in the proxy servers and the edge routers.
      * Delivering Web Content through a Geographically Distributed CDN



* + BGP (Border Gateway Protocol) based manual reconfiguration of the network routing used in the internet backbone has emerged as a powerful tool for defending local networks from DDoS attacks.
    - Autonomous System (AS): A network controlled by a single organization. An AS will have direct network connections with multiple other ASs through the internet backbone.
    - since the routing policies regarding the exchange of traffic between the peers are configured and updated manually, an instant routing policy change can re-route an ongoing DDoS attack through a longer route in which the DDoS traffic can be subject to greater filtering.
  + Mirai botnet are meant for infecting Internet of Things (IoT) devices.
    - IoT devices are default-insecure: their manufacturers ship them with hard-coded usernames and passwords that are left unchanged by the users.
    - IoT devices don’t possess the computational wherewithal for software maintenance and security-related updates.
    - Mirai botnet randomly scans IP address blocks looking for connections with IoT devices, connections that can be made with usernames and passwords programmed into Mirai code