Intro to Linux
Kernel Firewall
Linux Kernel Firewall

- Kernel provides Xtables (implemented as different Netfilter modules) which store chains and rules
  - x_tables is the name of the kernel module carrying the shared code portion used by all four modules that also provides the API used for extensions
  - Xtables is more or less used to refer to the entire firewall (v4,v6,arp,eb) architecture
- Different kernel modules and user space programs are currently used for different protocols
  - iptables applies to IPv4
  - ip6tables to IPv6
  - arptables to ARP
  - ebtables for Ethernet frames
- All require elevated privileges to operate and must be executed by root
- Not "essential binaries"
  - User must install in some distros
  - May reside in /sbin or /usr/sbin
Netfilter.org

- Home to the software of the packet filtering framework inside the Linux kernel
  - Software commonly associated with netfilter.org is iptables.
- Software inside this framework enables packet filtering, network address [and port] translation (NA[P]T) and other packet mangling
  - netfilter is a set of hooks inside the Linux kernel that allows kernel modules to register callback functions with the network stack.
  - iptables is a generic table structure for the definition of rulesets. Each rule within an IP table consists of a number of classifiers (iptables matches) and one connected action (iptables target).
  - netfilter, ip_tables, connection tracking (ip_conntrack, nf_conntrack) and the NAT subsystem together build the major parts of the framework.
- Main Features
  - stateless and stateful packet filtering (IPv4 and IPv6)
  - network address and port translation, e.g. NAT/NAPT (IPv4 only)
  - multiple layers of API's for 3rd party extensions
  - large number of plugins/modules kept in 'patch-o-matic' repository
- What can I do with netfilter/iptables?
  - build internet firewalls based on stateless and stateful packet filtering
  - use NAT and masquerading for sharing internet access
  - use NAT to implement transparent proxies
  - aid the tc and iproute2 systems used to build sophisticated QoS and policy routers
  - do further packet manipulation (mangling) like altering the TOS/DSCP/ECN bits of the IP header
Iptables

- Administration tool for IPv4 packet filtering and NAT
- Used to set up, maintain, and inspect the tables of IPv4 packet filter rules in the Linux kernel
- Several different tables may be defined
- Each table contains a number of built-in chains and may also contain user-defined chains
- Each chain is a list of rules which can match a set of packets
- Each rule specifies what to do with a packet that matches
- This is called a `target', which may be a jump to a user-defined chain in the same table.
- Targets
  - ACCEPT
    - let the packet through
  - DROP
    - drop the packet on the floor (think /dev/null)
  - QUEUE
    - pass the packet to userspace
  - RETURN
    - Stop traversing this chain and resume at the next rule in the previous (calling) chain
Predefined Chains

- **PREROUTING**
  - Packets will enter this chain before a routing decision is made

- **INPUT**
  - Packet is going to be locally delivered

- **FORWARD**
  - All packets that have been routed and were not for local delivery will traverse this chain
  - Requires IP forwarding to be enabled in kernel (same for routing)
    - echo 1 > /proc/sys/net/ipv4/ip_forward
    - /etc/sysct1.conf → net.ipv4.ip_forward = 1

- **OUTPUT**
  - Packets sent from the machine itself will be visiting this chain

- **POSTROUTING**
  - Routing decision has been made. Packets enter this chain just before handing them off to the hardware
There are currently three independent tables (which tables are present at any time depends on the kernel configuration options and which modules are present).

- **filter**
  - This is the default table (if no -t option is passed). It contains the built-in chains INPUT, FORWARD, and OUTPUT

- **nat**
  - This table is consulted when a packet that creates a new connection is encountered. It consists of three built-ins: PREROUTING, OUTPUT, and POSTROUTING.

- **mangle**
  - This table is used for specialized packet alteration. It consists of five built-ins: PREROUTING, INPUT, FORWARD, OUTPUT, and POSTROUTING.

- **raw**
  - This table is used mainly for configuring exemptions from connection tracking in combination with the NOTRACK target. It registers at the netfilter hooks with higher priority and is thus called before ip_conntrack, or any other IP tables. It provides the following built-in chains: PREROUTING and OUTPUT
The options that are recognized by iptables can be divided into several different groups. Way to many to list or cover

- **Commands**
  - These options specify the desired action to perform. Only one of them can be specified on the command line unless otherwise stated below. For long versions of the command and option names, you need to use only enough letters to ensure that iptables can differentiate it from all other options.
  - append, delete, insert, replace, list, list-rules, flush, zero, new-chain, delete-chain, policy, and rename-chain

- **Parameters**
  - parameters make up a rule specification (as used in the add, delete, insert, replace and append commands)
  - protocol, source, destination, jump, goto, in-interface, out-interface, fragment, and set-counters

- **Other options**
  - Verbose, numeric, exact, line-numbers, and modprobe
Match Extensions

iptables can use extended packet matching modules
- addrtype, ah, cluster, comment, connbytes, connlimit, connmark, conntrack, cpu, dccp, dscp, ecn, esp, hashlimit, helper, icmp, iprange, ipvs, length, limit, mac, mark, multiport, osf, owner, physdev, pkttype, policy, quota, rateest, realm, recent, sctp, set, socket, state, statistic, string, tcp, tcpmss, time, tos, ttl, u32, udp, and unclean

- These are loaded in two ways:
  - implicitly, when -p or --protocol is specified
  - or with the -m or --match options, followed by the matching module name
  - after these, various extra command line options become available, depending on the specific module
  - You can specify multiple extended match modules in one line, and you can use the -h or --help options after the module has been specified to receive help specific to that module.
Stateful Packet Inspection

A stateful firewall keeps track of the state of the network connections traveling across it. The firewall is programmed to distinguish legitimate packets for different types of connections. Only packets matching a known active connection will be allowed by the firewall; others will be rejected.

- The state match extension module, when combined with connection tracking, allows access to the connection tracking state for this packet.
  - NEW
    - A packet which creates a new connection.
  - ESTABLISHED
    - A packet which belongs to an existing connection (i.e., a reply packet, or outgoing packet on a connection which has seen replies).
  - RELATED
    - A packet which is related to, but not part of, an existing connection, such as an ICMP error, or (with the FTP module inserted), a packet establishing an ftp data connection.
  - UNTRACKED
    - A packet is not tracked at all, which happens if you use the NOTRACK target in raw table.
  - INVALID
    - A packet which could not be identified for some reason, does not correspond to any known connection. Generally these packets should be dropped.
Examples

- **Always Flush First**
  - `iptables -F`
  - `iptables -t nat -F`

- **Accept**
  - `iptables -A INPUT -p TCP -i eth0 --sport 20:23 -j ACCEPT`

- **Log, reject, drop**
  - `iptables -A INPUT -i eth0 -m limit --limit 5/minute -j LOG --log-prefix "iptables: "`
  - `iptables -A INPUT -i eth0 -j REJECT --reject-with icmp-host-unreachable`
  - `iptables -A INPUT -i eth0 -j DROP`

- **Stateful**
  - `iptables -P INPUT DROP`
  - `iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT`
  - `iptables -A INPUT -i eth0 --dport 22 -m state --state NEW -j ACCEPT`

- **NAT**
  - `iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE`
  - `iptables -t nat -A POSTROUTING -o tun0 -s x.x.x.x -j SNAT --to x.x.x.x`

- **PAT**
  - `iptables -t nat -A PREROUTING -i eth0 -p TCP --dport 80 -j DNAT x.x.x.x:x`

- **Port redirection**
  - `iptables -t nat -A PREROUTING -p TCP --dport 25 -j DNAT --to :125`
References

- iptables man page
- http://www.netfilter.org/