NETWORK CONFIGURATION AND SERVICES

route add default gw 192.168.0.1

/etc/init.d/apache restart

NETWORK CONFIGURATION

- There are two main approaches to configuring a machine for network access:
  - Static configuration
  - Dynamic configuration
- Static configuration uses set parameters for the configuration, which is known by the machine and the network and never changes. Generally used with servers.
- Dynamic configuration configures network machines on the fly, where a service on the network provides all configuration parameters to a machine when it joins the network. Generally used with workstations.

DYNAMIC CONFIGURATION

- Dynamic configuration is the easiest to use.
- The machine just needs to set up it’s interfaces with the DHCP protocol.
- DHCP: Dynamic Host Configuration Protocol.
- A <u>lease</u> is obtained from the DHCP server, providing all network configuration details for the client. The lease expires after some amount of time and is renewed by the client to maintain network access.

STATIC CONFIGURATION

- Static configuration requires four configuration parameters in order to allow full network functionality:
  - IP Address
  - Netmask
  - Default Gateway or Router
  - DNS Server(s)

DNS?

- Domain Name Service: This is the glue between network names and IP addresses.
- Remember: Humans like names, computers like numbers. DNS is a service like so many others, mapping names to numbers and numbers to names. Mostly a convenience.
- Also provides for email functionality, geographic load balancing and limited service failover capabilities.

STATIC CONFIGURATION

- The first two components of static configuration are IP address and netmask.
- These provide LAN-level access
- `ifconfig`: Network Interface configuration tool
- Basic idea:
  - `ifconfig eth0 192.168.0.100 netmask 255.255.255.0`
- Live examples are good!

GATEWAYS

- The third configuration parameter is the default gateway.
- Provides access to *inter-networking*, or moving from just the local LAN to other LAN's
- `route`: Kernel routing table tool
    - Displays and manipulates network routing table
    - `route add default gw 192.168.0.1`
- More live examples!

DNS SERVERS

- Final piece of configuration information.
- List of one or more IP addresses which provide the DNS service, allowing name to IP address mapping
- Very simple to configure. Add `nameserver` lines to `/etc/resolv.conf`:
  - `nameserver 192.168.7.15`
- Also consider `/etc/nsswitch.conf`

STATIC CONFIGURATION

- Once all four pieces of information are configured on the system, full network service will be available.
- Best practice:
  - Configure IP address and netmask. Check LAN connectivity.
  - Configure default gateway. Check intra-LAN connectivity.
  - Configure DNS: Check name resolution.

ONE MORE THING...

- ifconfig and route directly manipulate the running kernel, and are not permanent changes to the system. After a reboot, changes will be lost.
- To make IP address, netmask and gateway changes permanent, you have to edit two configuration files:
  - `/etc/sysconfig/network-services/ifcfg-eth0`
  - `/etc/sysconfig/network`
- Let's look at these files on our systems...

EXERCISES

- Check your current IP address, default route and DNS servers.
- Restart networking services using the proper init script.

TCP WRAPPERS

- TCP Wrappers was originally written to provide host based access control for services which did not already include it.
- It was one of the first “firewalls” of a sort. :)
- Before you can set up tcp_wrappers on a service, you have to check if the service supports it...

CHECKING TCP WRAPPER SUPPORT

- Determine which binary the application runs as. Check `init` script or:

```
# which sshd

/usr/sbin/sshd
```

- Check for `libwrap` support in the binary.

- If you see `libwrap` support in the output, then you can configure access to the service with tcp_wrappers.

```
# ldd /usr/sbin/sshd | fgrep wrap

libwrap.so.0 => /usr/lib/libwrap.so.0 (0x009c5000)
```

TCP WRAPPER OPERATION

- If an application is compiled with support for `tcp_wrappers`, that application will check connection attempts against the `tcp_wrappers` configuration files:
    - `/etc/hosts.allow`
    - `/etc/hosts.deny`

TCP WRAPPER OPERATION

- These files are parsed in the following order:
  - The file `/etc/hosts.allow` is consulted. If the configuration of this file permits the requested connection, the connection is immediately allowed.
  - The file `/etc/hosts.deny` is consulted. If the configuration of this file does not permit the requested connection, the connection is immediately refused.
  - If the connection is not specifically accepted or rejected in either file, the connection is permitted.

TCP WRAPPER CONFIGURATION

- The basic syntax for these files is:
    - `<daemon>: <client>`
- For example, to disable `ssh` connections from `192.168.2.200`, add this line to `/etc/hosts.deny`:
    - `sshd: 192.168.2.200`

FIREWALLS

- A firewall is a mechanism for defining rules about valid and invalid traffic, which then directs what to do with the traffic
- In Linux, the firewall implementation is called `iptables`.
- `iptables` is a very powerful firewall system, providing extensive flexibility in rule definition and actions.

IPTABLES

- IPTables works at the kernel level, just above the network drivers, to provide several useful features.
  - Extremely powerful and flexible Layer 2 filtering engine.
  - NAT support
  - Port forwarding
  - And many more

IPTABLES RULE MATCHING

- The IPTables configuration is parsed from top to bottom.
- IPTables will respond based on the first match that it finds.
- If there is no specific match, the chain policy will apply.

IPTABLES TOOLS

`iptables`:	View/modify current firewall rules
`iptables-save`:	Script to save current firewall rules for use with iptables-restore
`iptables-restore`:	Restores iptables-save format firewall rules - useful to set up firewalls at boot

IPTABLES RULES

- When creating a new rule, considerations include:
  - What chain should the rule apply to?
  - What is the traffic pattern to look for?
  - What should happen with the traffic?

IPTABLES CHAINS

- **`INPUT`**
    - This chain is responsible for filtering traffic destined for the local system.
- **`OUTPUT`**
    - This chain is responsible for handling outbound traffic.
- **`FORWARD`**
    - This chain is responsible for controlling traffic routed between different interfaces.

IPTABLES RULES

- Below are a few examples of possible IPTables match criteria:

incoming interface	`-i`
protocol	`-p`
source ip address	`-s`
destination ip address	`-d`
destination port	`--dport`

IPTABLES RULES

- Finally, some examples of what to do with traffic when matched:

`DROP`	Do not deliver, do not respond
`REJECT`	Do not deliver, send reject notice
`ACCEPT`	Deliver
`LOG`	Just log the packet

IPTABLES RULES

- So to summarize the syntax:
    - `iptables`
  - What chain should the rule apply to?
    - `-A INPUT`
  - What is the traffic pattern to look for?
    - `-s 192.168.2.100`
  - What should happen with the traffic?
    - `-j REJECT`

LAB

1. Using `iptables`, configure your server to NOT accept SMTP connections from the `192.168.1.0/24` network, EXCEPT for the `192.168.1.2` host.

NETWORK SERVICES

- There are hundreds, even thousands of different network services out there. And each individual service might have one or a dozen plus applications which can provide the service.
- The big ones, and the ones overviewed in this course:
  - HTTP, SMTP, SSH, FTP, MySQL

HTTP

- Hypertext Transfer Protocol: Used to ship webpages and associated files across the network.
- Popular servers: Apache, IIS, lighttpd

SMTP

- Simple Mail Transfer Protocol: Delivers email messages
- Popular servers: qmail, sendmail, postfix, exchange

SSH

- Secure Shell: Encrypted remote shell access
- Server: OpenSSH
- Consider: ssh keys, known hosts, authorized hosts

FTP

- File Transfer Protocol: Used to move files around the network
- Popular Servers: wu-ftpd, vsftpd

MYSQL

- MySQL: Extremely powerful, open-source relational database management system
- MySQL is the server, and the only one, as this service is completely defined and implemented by the application

NTP

- The Network Time Protocol is a very useful and accurate method to keep your system clock synchronized with time servers around the world. This is important because:
  - Timestamps in log files across machine will line up, allowing for proper analysis and comparison
  - Cron jobs run at the right time
  - Knowing the correct time just makes for a happy server

XINETD

- `xinetd` is the extended internet services SUPER daemon. :)
- This service acts as a super daemon by listening on key ports for certain types of requests.
- When a request is received, `xinetd` starts the appropriate service and then hands of the request so that it can be handled correctly.
- `xinetd` is configured in `/etc/xinetd.conf`, the services that it controls are configured in `/etc/xinetd.d/`

GPG

- gpg: GNU Privacy Guard
  - Basically, GNU implementation of PGP
  - Public Key Infrastructure encryption

```
slideshow.end();
```