#Routers and Firewalls [ETC: 1.5 Hours] This section will be about configuring a router to correctly forward data and creating a ruleset to be deployed on a firewall or router that will protect the network.

Routers allow communication between computer networks. In order to access the internet and allow remote hosts to access our services we will be allowing communication to hosts that are public and out of our control. In this case we would like the router or another piece of hardware or software to protect our network from unwanted traffic. Firewalls and some rotuers can inspect network traffic to varying degrees and block malicious or unwanted traffic based on a set of rules. ##Network Basics Required knowledge for this section:

- Basic understanding of the TCP/IP stack
- Using a command line interface

##Network Layout
A typical network layout will have:

- A gateway router and a firewall, these can be separate or combined appliances or software
- One or more workstation subnetworks
- One or more server subnetworks

//Insert diagram of Network Layout

Remote threats to both the workstations and the servers exist but they will need to be protected in different ways on the network.
##Cisco Hardware If the competition has specific networking hardware it will very likely be Cisco. Cisco creates both the hardware and the software for their lines of routers, switches, and firewalls. ###Connecting to Cisco Hardware Your first connection to a piece of cisco hardware will almost always be done through a rollover cable connected to the cisco's console port and to your computers serial port. You will need to use a terminal emulator that can talk over the serial port. Putty and WinRS are both windows programs that will do this. On linux, minicom is a good choice. Whatever program you choose some configuration may be nessecary:

```
bps (baud)  : 9600
Data Bits   : 8
Parity      : None
Stop Bits   : 1
Flow Control: None
```

For some terminal emulators like minicom, these configuration options may be presented as:

```
9600-8N1
```

To make the connection in putty:

- Make sure your router is connected to your serial port and turned on.
- Open a new putty configuration
- Choose serial then enter your Serial Line (likely COM1 but depends on hardware configuration) and a Speed of 9600
- Open the connection and send a return, you should see a cisco prompt.

###Help I Dont Know the Password Information on recovering a password for a cisco router can be found here. ###Basic Router Configuration Hardening an IOS based router is fairly simple but must be done through the command line interface. ####Securing Access and Creating a Password Creating a password will prevent other people from being able to access your router. In addition to creating a password we will want to encrypt that password so that it is not sent as plaintext over the network if you need to do any more configuration. You may chose to not enable telnet login and always configure through the console port, this is more secure but also more work, it's your descision.
When you first connect to a cisco router you are in unpriviliged mode with a prompt:

```
Router>
```

Giving the enable command will take you to a priviliged session:

```
Router>enable
Router#
```

From the privliged session you can enter configuration mode using the command config terminal:

```
Router#config terminal
Router(config)#
```

At this point you will be able to configure the routers interfaces and general options such as adding a password.
To create and encypt a password we will issue two commands:

```
enable secret [yourpassword]
service password-encryption
```

Additional commands relevant to addding and encrypting passwords can be found here. ####Disabling Non-essential Services Some, but not all, cisco routers and switches have additional services such as a web interface or tftp file transfer options. We will want to make sure all of these are disabled by entering the following commands after successfully logging into the router:

```
    !turn off possibly exploitable small, non-essential services
    no service tcp-small-servers
    no service udp-small-servers
    no ip finger
    no service finger !older ios
    no ip bootp server         i
    no service pad
    no boot host
    no boot network
    no boot system

    !disable http interface
    no ip http server
    no ip http secure-server

    !disable remote loading of config files
    no service config

    !disable cdp
    no cdp run !global
    no cdp advertise-v2
    no cdb enable !interface

    no ip domain-lookup

    !disable snmp cleartext (and completely actually)
    no snmp-server community
    no snmp-server enable traps
    no snmp-server system-shutdown
    no snmp-server
```

under line 6, "no ip bootp server i". Should the i be there?

##Defining Firewall/ACL Rules There are many different syntaxes that firewalls are written in but almost all of them follow the same paradigm and general arguments. An example acl or firewall statement:

```
        <action> <source> <source subnet> <destination> <destination subnet>
<protocol> [port]
```

For example:

```
    block 192.168.1.0 255.255.255.0 192.168.2.0 255.255.255.0 tcp 80
```

Denys web traffic (http on port 80) from the 192.168.1.0/24 network to the 192.168.2.0/24 network

Most syntaxes allow some shortcuts to be taken when writing firewall or access control lists such as:

```
        permit any 192.168.2.123 ssh
```

Could allow any address to access ssh (tcp port 22 by default) services on the 192.168.2.123 host

Most firewall or access list implementations will parse a ruleset top down such that for the ruleset:

```
    deny any any
    permit 192.168.1.0 255.255.255.0 any
```

The 192.168.1.0 network will still not be able to pass any packets because the 'deny any any' rule is parsed before the permit rule in the list. ####Example Cisco ACL

```
    permit icmp any any
        !--- Allow all ICMP such as pings from anywhere to anywhere
    permit tcp 192.168.1.0 255.255.255.0 any eq 80
    permit tcp 192.168.1.0 255.255.255.0 any eq 443
        !--- Allow our local network to access any address on ports equal to 80 or
443 for web traffic
    permit ip 192.168.1.0 255.255.255.0 192.168.2.0 255.255.255.0
        !--- Allow our local network to access our server network services on any ip
protocol
    permit tcp any 192.168.1.0 255.255.255.0 gt 1023 established
        !--- Allow traffic from anywhere to our local network on any port greater
than 1023 so long as the connection is established
    deny any any
        !--- Catch all and deny remaining traffic
```

###Access List Requirements by Role ####Workstations Workstations often require only basic outbound access to known services but require dynamic ports inbound in order to communicate. For example a windows 7 workstation may need very few exceptions in the firewalls:

```
        permit <workstation_subnet> any tcp 80
        permit <workstation_subnet> any tcp 443
        deny any any
```

The above example would allow traffic on ports 80 & 443 (often http and https traffic, including windows update for security updates!) out from the workstation computers but will not allow any malicious program on the workstations to reach out on another port. The workstations will use these open ports for outbound web traffic, however, we also need to allow traffic to return to the computers so that web pages can actually be loaded. Traffic comes back from the servers on dynamic ports so we can't simply say "let 80 and 443 flow both ways", instead the firewall or router can examine a TCP stream and decide if it is "established" meaning traffic is already moving in at least one direction (because we have no rules allowing traffic in yet, this means we must have started the connection) and add an inbound rule along the following lines:

```
        permit any <workstation_subnet> tcp established
        deny any any
```

This allows any tcp traffic on any port that is an established stream to be forwarded to the workstation computers, this is ideal because we can now browse the web but are protected from malicious programs trying to start connections on other ports and from many possible vulnerabilities that may still be open on each workstation.
####Servers **Note** I need to do a little further research about established connections out from servers, this could be really strong for blocking malware that already has persistence but I don't want to give wrong info so I'll leave this blank for now. ##Firewall Software Some of the possible open source software you may run into includes:

- IPFilter
- ipfw
- [m0n0wall]http://m0n0.ch/wall/
- Smoothwall

Some of these are configured through the command line and some of them are configured through a gui but the concept of ACLs remains the same on each of the ones listed. There will be a list rules that each packet is checked against in descending order, the first rule that matches is used to decide what to do with each packet. Below we will use IPFilter on our Ubuntu VM to demonstrate a very simple configuration.

##Exercise Start the unix vm called ccdc-firewalls. Once your vm is up you should be able to browse to its IP address on your host computer and view a webpage, you will know if it's working.
On the VM you can edit the IPFilter configuration file stored at:

```
/etc/ipf/acl.conf
```

At the top of the file add the line:

```
block any any tcp 80
```

Issuing the command service ipfilter restart will reload the configuration files

```
service ipfilter restart
```

This blocks any requests on port 80 tcp going in or out. When you try to navigate to the page that worked earlier you should see that it infact is now blocked! .