IP Address Configuration

Debian

1. Edit `/etc/network/interfaces`
2. `systemctl restart networking`

Configuration

```
iface [interface] inet static
      address [ip]
      netmask [netmask]
      gateway [gateway]
      dns-nameserver [nameserver]
```

Sample Configuration

```
iface ens18 inet static
      address 192.168.14.5
      netmask 255.255.255.0
      gateway 192.168.14.1
      dns-nameserver 192.168.14.7
      dns-nameserver 172.20.0.7
```

Ubuntu

1. Edit `/etc/netplan/01-network-manager-all.yaml`
2. `netplan apply`

Configuration

```
network:
   ethernets:
      [interface]:
         addresses:
            - [ip]/[CIDR]
         gateway4: [ipv4 gateway]
         nameservers:
            addresses: [[nameserver 1], [nameserver 2], […]]
```

Sample Configuration

```
ethernets:
   ens18:
      addresses:
         - 192.168.14.5/24]
      gateway4: 192.168.14.1
      nameservers:
         addresses: [192.168.14.7, 172.20.0.7]
```

CentOS

1. Edit `/etc/sysconfig/network-scripts/ifcfg-`[device]
2. `systemctl restart network`

Configuration

```
BOOTPROTO=static
ONBOOT=yes
IPADDR=[ip]
NETMASK=[netmask]
GATEWAY=[gateway]
DNS[1/2/3/…]=[nameserver]
ZONE=[zone]
```

Sample Configuration: `/etc/sysconfig/network-scripts/ifcfg-ens18`

```
BOOTPROTO=static
ONBOOT=yes
IPADDR=192.168.14.5
NETMASK=255.255.255.0
GATEWAY=192.168.14.1
DNS1=192.168.14.7
DNS2=172.20.0.7
ZONE=internal
```

Firewall Configuration (Firewalld)

Don't forget to run `echo 1 > /proc/sys/net/ipv4/ip_forward`
`systemctl restart network`

Zone Commands

- `firewall-cmd --list-all-zones` ⇒ shows all zone information
- `firewall-cmd --get-zones` ⇒ only shows zone names
- `firewall-cmd --list-all --zone=`[zone] ⇒ shows info for specific zone
  or `firewall-cmd --info-zone=`[zone]
- `firewall-cmd --new-zone=`[zone] `--permanent` ⇒ creates a new zone
- `firewall-cmd --set-default-zone=`[zone] ⇒ sets default zone
  - Default zone is used for everything that's not assigned to another zone

Rule Commands

- `firewall-cmd`
  - `--zone=`[zone] → if this is not specified, it will modify the default zone
  - `--permanent` ⇒ persist on service restart
  - [rules]

Rule	Description	Command Option (Flag)
Interface	The interface assigned to this zone	`--change-interface=`[interface] ⇒ assign interface to this zone **Note:** you can do this instead of adding `ZONE=`[zone] to the CentOS IP configuration file.
Source	Whitelist (accept connections from) IP addresses	`--add-source=`[ip] `--remove-source=`[ip]
Target	How to handle packets that don't match any other rules	`--set-target=`[accept/reject/drop]
Service	Services running on *this* machine that are accessible by this zone	`--add-service=`[service] `--remove-service=`[service] **Note:** use `firewall-cmd --get-services` to list available services.
Port	Ports running on *this* machine that are accessible by this zone. Use when a service is not available	`--add-port=`[port]`/`[`tcp`/`udp`] `--remove-port=`[port]`/`[`tcp`/`udp`]
Forward Port	Ports to be forwarded to *other* machines that are accessible by this zone	`--add-forward-port=port=`[source port]`:proto=`[`tcp`/`udp`]`:toport=`[destination port]`:toaddr=`[destination address] `--remove-forward-port=`[same options]
Masquerade	Allow masked outbound connections on this zone (useful for external)	`--add-masquerade` `--remove-masquerade`

Always load changes to rules with `firewall-cmd --reload`

Sample Port Forwards

- **SSH:** `firewall-cmd --zone=external --permanent --add-forward-port=port=22:proto=tcp:toport=22:toaddr=192.168.14.5`
- **HTTP:** `firewall-cmd --zone=external --permanent --add-forward-port=port=80:proto=tcp:toport=80:toaddr=192.168.14.5`
- **HTTPS:** `firewall-cmd --zone=external --permanent --add-forward-port=port=443:proto=tcp:toport=443:toaddr=192.168.14.5`

- **DNS:** `firewall-cmd --zone=external --permanent --add-forward-port=port=53:proto=udp:toport=53:toaddr=192.168.14.5`

Note: DNS uses UDP not TCP!

Secure Shell (SSH)

Note: There are two types of keys:
- Host key for client to authenticate server device
- User keys for server to authenticate client user

Client

- `ssh` [user]`@`[ip] ⇒ password-based authentication
- `ssh -i` [private key file] [user]`@`[ip] ⇒ key-based authentication

Important Files
- `/etc/ssh/ssh_config` ⇒ main configuration file
- `/etc/ssh/ssh_config.d` ⇒ directory with additional configuration files

- [user home]`/.ssh/known_hosts` ⇒ keys for known devices

- [user home]`/.ssh/id_`[algorithm] ⇒ private key for user
  - Used to authenticate to other machines
- [user home]`/.ssh/id_`[algorithm]`.pub` ⇒ public key for user
  - Gets stored in server's authorized_keys file to authenticate client

Server

- `systemctl [start/stop/status] sshd`

Important Files
- `/etc/ssh/sshd_config` ⇒ main configuration file
- `/etc/ssh/sshd_config.d` ⇒ directory with additional configuration files
- [user home]`/.ssh/authorized_keys` ⇒ keys that can be used to connect to this machine as this user

Generating Keys

Replace Host Key
- `ssh-keygen -t` [algorithm] `-f /etc/ssh/ssh_host_`[algorithm]`_key`

Example: `ssh-keygen -t ecdsa -f /etc/ssh/ssh_host_ecdsa_key`

Add Key For User

Option 1
- `ssh-keygen -t` [algorithm] `-f` [key]

- Copy [key]`.pub` content to [user home]`/.ssh/authorized_keys`
- Make sure [user home]`/.ssh` owned by user has 700 privileges
- Make sure `authorized_keys` owned by user has 644 privileges

Example
- `ssh-keygen -t ecdsa -f id_bob`
- `cp id_bob.pub /home/bob/.ssh/authorized_keys`
- `chown bob:bob /home/bob/.ssh`
- `chmod 700 /home/bob/.ssh`
- `chown bob:bob /home/bob/.ssh/authorized_keys`
- `chmod 644 /home/bob/.ssh/authorized_keys`

Option 2 *recommended*
- `ssh-keygen -t` [algorithm] `-f` [key]
- `ssh-copy-id -i` [key] [user]`@`[ip]

Example
- `ssh-keygen -t ecdsa -f id_bob`
- `ssh-copy-id -i id_bob bob@localhost`

Requiring Key-Based Authentication

1. Edit the following lines in `/etc/ssh/sshd_config`:

```
ChallengeResponseAuthentication no
PasswordAuthentication no
UsePAM no
```

2. (Optional) To disable root login, edit the following line in `/etc/ssh/sshd_config`:

```
PermitRootLogin no
```

3. `systemctl reload sshd`
   or `/etc/init.d/ssh reload`

Web Server (HTTP/HTTPS)

Configuration File
- `/etc/apache2/apache2.conf` ⇒ Ubuntu and Debian
- `/etc/httpd/conf/httpd.conf` ⇒ CentOS
  - Additional configuration files in `/etc/httpd/conf.d`

Note: This might be worth looking through for hardening.

Using an SSL Certificate
1. Install the mod_ssl package if not already installed.

2. Generate and (optionally) sign SSL key.
   How to do this depends on context. You can generate your own unsigned key with `openssl`, use an online site, or a server that generates the key for you.
3. Locate the secured site configuration file.
   - On Ubuntu/Debian, check the `/etc/apache2/sites-enabled` directory
   - On CentOS, check for `/etc/httpd/conf.d/ssl.conf`
4. Add the following lines to the secured site configuration file:

```
<VirtualHost *:443>
DocumentRoot /var/www/html
ServerName [domain name]
SSLEngine on
SSLCertificateFile [certificate].crt
SSLCertificateKeyFile [key].key
SSLCertificateChainFile [chain].crt
</VirtualHost>
```

Certificate ⇒ primary SSL certificate file
Key ⇒ private key file received when generating Certificate Signing Request (CSR) file
Chain (optional) ⇒ certificate file received from SSL certificate issuing authority

5. `apachectl configtest`
6. `systemctl restart [apache2/httpd]`

Domain Name System (DNS)

Installing and Configuring Bind

Note: The following instructions are adapted from a variety of sources. Bind9 might also be installed as named! In this case, package names will be different.

1. Switch to root user for easier terminal use (root user does not require `sudo`).
   a. `sudo su`
2. Install the required libraries and dependencies.
   a. `apt update`
   b. `apt install bind9 bind9utils bind9-doc dnsutils`
3. Create the DNS zones.
   a. Edit the `/etc/bind/named.conf.local` file.
      i. Example: `nano /etc/bind/named.conf.local`

Add the following forward zone information:

```
zone "[base URL]" IN {
      type master;
      file "/etc/bind/[forward lookup file]";
```

```
     allow-update { none; };
};
```

Sample forward zone information:

```
zone "cyberhawks.org" IN {
     type master;
     file "/etc/bind/forward.cyberhawks.org";
     allow-update { none; };
};
```

Add the following reverse zone information:

```
zone "[local IP in reverse order discounting last two digits].in-addr.arpa" IN {
     type master;
     file "/etc/bind/[reverse lookup file]";
     allow-update { none; };
};
```

Sample reverse zone information:

```
zone "0.0.10.in-addr.arpa" IN {
     type master;
     file "/etc/bind/reverse.cyberhawks.org";
     allow-update { none; };
};
```

4. Create the forward zone lookup file.
    a. `cp /etc/bind/db.local /etc/bind/`[forward lookup file]
        i. Example: `cp /etc/bind/db.local /etc/bind/forward.cyberhawks.org`
    b. Edit the forward lookup file.
        i. Example: `nano /etc/bind/forward.cyberhawks.org`

Update the following line of code:

```
@        IN       SOA      ns.[base URL]. root.[base URL]. (
```

Example:

```
@        IN       SOA  ns.cyberhawks.org. root.cyberhawks.org. (
```

Enter the forward DNS records.

Record types:
SOA ⇒ Start of Authority
NS ⇒ Name Server (for name servers)
A ⇒ A Record (map URL to IP)
MX ⇒ Mail for Exchange (for mail servers)
CNAME ⇒ Canonical Name (aliases)

Sample DNS record information:

```
; name server
@          IN         NS         ns.cyberhawks.org.
ns         IN         A          10.0.0.11

; web server
@          IN         A          10.0.0.4
www        IN         CNAME      @
```

5. Create the reverse zone lookup file.
    a. `cp /etc/bind/db.127 /etc/bind/`[reverse lookup file]
        i. Example: `cp /etc/bind/db.127 /etc/bind/reverse.cyberhawks.org`
    b. Edit the reverse lookup file.
        i. Example: `nano /etc/bind/reverse.cyberhawks.org`

Update the following line of code:

`@        IN       SOA      ns.`[base URL]`. root.`[base URL]`. (`

Example:

```
@        IN       SOA  ns.cyberhawks.org. root.cyberhawks.org. (
```

Enter the reverse DNS records.

Record types:
SOA ⇒ Start of Authority
NS ⇒ Name Server (for name servers)
PTR ⇒ Pointer (reverse of A and MX pointing to the domain name)

Sample DNS record information:

```
; name server
@          IN         NS         ns.cyberhawks.org.
11         IN         PTR        ns.cyberhawks.org.

; web server
4          IN         PTR        cyberhawks.org.
```

Checking Bind Configuration Syntax

1. Check the configuration file for any errors.
    a. `named-checkconf`
2. Check the zones for any errors.
    a. `named-checkzone` [base URL] `/etc/bind/`[lookup zone]
        i. Example: `named-checkzone cyberhawks.org /etc/bind/forward.cyberhawks.org`

Starting Bind

1. `systemctl enable [bind9/named]`
2. `systemctl start [bind9/named]`

Using DNS Name Resolution on a Client Machine

Note: This varies depending on the architecture of the client machine. The following should work for any standard Linux OS.

1. Update the default DNS nameserver configuration.
    a. Edit the `/etc/resolv.conf` file.
        i. Example: `nano /etc/resolv.conf`

Add the following lines of code:

```
search [base URL]
nameserver [public/private IP address of DNS name server]
```

Sample configuration:

```
search cyberhawks.org
nameserver 40.122.68.21
```

2. You should now be able to access any record provided by the server that is within the [base URL] zone.
    a. Example: http://www.cyberhawks.org opens CyberHawks webpage on LAN machine 10.0.0.4.

Realtime Defence

The Plan

https://github.com/RedefiningReality/Linux-Defense-Scripts

1. Look for shells other than yours with `who -l`
2. Check process associated with each shell with `ps eaf --forest`
    a. This view (forest) of processes shows nested shells (shells that call other shells) which is common for privilege escalation. Remember nested shells so you can check history file of each user to determine both initial attack vector and escalation attack vector
3. Send attacker trolly messages for the memes
    - `wall "[message]"` ⇒ broadcast message to all users
    - `wall -n "[message]"` ⇒ broadcast message without banner
    - `write [user] [optional: tty]`
4. Kick attacker out with `kill -9 [process id]`
5. Determine how they got in:
    a. Check command history file for users compromised
    b. If history file doesn't exist, check log files

Useful Defence Commands

- `who` ⇒ list terminal sessions
- `who -l` ⇒ list terminal sessions with process ID
- `ps aux or ps eaf --forest` ⇒ list processes
- `kill -9` [pid] ⇒ kill process

- `apt install` [package]
- `apt --reinstall install` [package]
- `apt remove` [package]
  or `apt purge` [package]

- `systemctl [start/stop/status]` [service]
  or `service` [service] `[start/stop/status]`

Undo the silly things attackers do just to mess with you:

- `chattr -i` [file] ⇒ make file mutable if it's immutable (making it unchangeable)
- `systemctl unmask` [service] ⇒ unmask service if it is masked (making it unusable)

Configuring Command History

Note: I would not do this in a production environment because bash is slower than dash, but this could be useful for cyber defence competitions. Also, keep in mind if the attackers figure out what I've done here, they could remove their command history, rendering this useless.

1. Change default shell to bash (bash stores command history) in one of the following ways:
   - Edit `/etc/passwd` ⇒ *recommended* you can also remove shells for users that don't need it by setting it to nologin as necessary
   - `usermod -s /bin/bash` [user] for each user
   - `dpkg-reconfigure dash` and answer `no` ⇒ dash is no longer default shell
   - `ln -s /bin/bash /bin/sh.bash ; mv /bind/sh.bash /bin/sh` ⇒ only replaces `/bin/sh` with `/bin/bash` so not advisable
2. Add the following lines to `.bashrc` for each user with login shell:
   a. `export HISTFILE=`[location] ⇒ changes history file location – remember that you can get username with `$USER`
   b. `shopt -s histappend` ⇒ appends commands rather than overwriting
   c. `export PROMPT_COMMAND="history -a ; history -r"` ⇒ writes commands to history file immediately after running command rather than on exit
3. Additional options (add to `.bashrc`):
   a. `export HISTSIZE=`[size] ⇒ # of command to store in memory
   b. `export HISTFILESIZE=`[size] ⇒ # of commands to store in history file

Log Files

Note: You can read the end of any of these with `tail` [logfile] or `tail -n` [number of lines] [logfile]

<u>Log File Locations</u>

- `/var/log/messages` ⇒ system messages
  or `/var/log/syslog`
- `/var/log/auth.log` or `/var/log/secure` ⇒ all login attempts
- `/var/log/faillog` ⇒ failed login attempts
- `/var/log/cron` ⇒ cron jobs
- `/var/log/apache2/access.log` ⇒ web requests
  or `/var/log/httpd/access.log`
- `/var/log/mysqld.log` ⇒ SQL log