Linux Academy

Study Guide

Linux Professional Institute Certification Level 2 Exam 2

Contents

Prerequisites

Linux - CentOS 6, CentOS 7 (or Other SysVInit or Systemd Distribution)

Topic 207 - Domain Name Server

207.1 Basic DNS Server Configuration (DNS Overview)

- DNS (Domain Name Service)
  - used to provide translation of IP addresses to names and is part of the TCP/IP standards that provide this type of functionality
  - additional name resolution protocols are LDAP and NIS, although DNS is unique in that is provides ONLY name resolution (unlike the other two)
- `/etc/resolv.conf`
  - the client configuration file for DNS resolution, directives include:
    - `nameserver [IP Address]` • one or more lines indicating the IP address of a server configured to respond to DNS queries from the client (typically up to three nameserver values are supported, order of appearance is the order they will be queried)
    - `search [domain]` • the default domain that will be searched for member names
    - `domain [domain]` • the default domain that all names will be assumed to be a member of unless specified by a FQDN
- root servers
  - these servers are the "end point," where DNS queries are passed off to by all other DNS servers
  - currently, there are 13 root server entries (described by default in the BIND zone files - see configuration further down)
- key terms
  - **Domain name** • valid names registered in a DNS server (they can be public, private or both)
  - **Top-level domain** • referred to by the "." (dot) character in a domain, the final portion of a domain name (i.e. .com, .net, .int, .gov, .edu, .mil - the original top level domains)
  - **FQDN (fully qualified domain name)** • the host name followed by the domain it belongs to (i.e. user1.mylabserver.com is the FQDN for one of the Linux Academy lab servers with the host name 'user1')

- **Subdomain** • part of the larger domain (for example, dev.mylabserver.com, qa.mylabserver.com, prod.mylabserver.com - each subdomain indicating the environment it represents)
- **Zone files and records** • this file stores the translation from IP to name and a record is the individual line that represents one such translation
- **caching name server** • a DNS server that returns address information received from another DNS server, used to speed DNS queries by caching those results for a period of time
- **TTL (time to live)** • the amount of time a cached DNS record will be returned before a query is sent back out to obtain a fresh copy (generally by default, it is 24 hours but can be set to any value desired, see subsequent BIND server setup later)
- **DNS forwarder** • simply forwards DNS requests from one network to another (often internal network to external network DNS)
- **Forward lookups** • IP to domain name (default DNS behavior)
- **Reverse lookup** • domain name to IP (most DNS servers provide in addition to forward lookups)
- **BIND (Berkley Internet Name Domain)** • most common DNS server, specific configuration tested during LPIC-2 Exam 2
- **dnsmasq** • additional DNS server that functions as a forwarding DNS server and/or for DHCP
- **PowerDNS** • load balanced DNS server (service) that was originally proprietary but later open sourced
- **djbdns** • developed as a more secure alternative to the BIND DNS server, more popular on Debian/Ubuntu systems

207.1 Basic DNS Server Configuration (BIND Configuration and Setup - Caching Name Server)

- **BIND9** • DNS server for LPIC-2 configuration
  - package on each distribution - bind, bind-utils
- `/etc/named.conf`
  - primary configuration file for the BIND server
  - read by the BIND service on start (BIND service named)
  - most settings will NOT need to be changed
  - the default settings are for a caching name server (part of the LPIC-2 requirements), simply starting the default configuration with the ‘named’ service will start a fully configured and functional caching only DNS server

- key configuration items
  - `listen-on` • port and interface(s) to listen on (semicolon delimited list)
    - **Example** • `listen-on port 53 { 127.0.0.1; 10.1.0.100; };`
      - would listen on the localhost IP and 10.1.0.100 over port 53 for DNS requests
  - **Directory** • by default, will be `/var/named`, but can be defined to any location (taking SELinux into consideration) for the working directory of the server (except for chroot jails, covered in a later topic)
  - `dump-file` • directory of the file created with the `rndc dumpdb` command
  - `allow-query` • indicates the systems the server will respond to (semicolon delimited list of allowed/disallowed systems)
    - **Example** • `allow-query { localhost; !10.1.0.100; 10.1.0/24 };`
      - will allow queries from localhost, and the entire 10.1.0/24 network EXCEPT 10.1.0.100
  - `dnssec-enable` • set to yes will enable the DNSSEC (Secure DNS Extension), for securing/authenticating DNS data (again, covered in a later topic in more detail)
  - `dnssec-validation` • enables the user of managed keys (trusted)
  - `bindkeys-file` • the file used if the `dnssec-lookaside` directive is set to auto
  - `managed-keys-directory` • the directory used to store the managed keys in a secure DNS server configuration
  - `session-keyfile` • contains the Transaction Signature session key (TSIG), used for authenticated updates to DNS (covered in more detail in a later topic for securing a DNS server)
  - `logging` • allows the definition of logging locations as well as the level of messages that are written (called severity)
  - `zone` • although not an essential setting for a caching nameserver, will specify the type and location of zone files for a hosted DNS service
  - `include` • specifies additional files (containing other settings) that can be included when the service is started (be careful about placement since its location will determine the exact place the included files rules are inserted)

207.1 Basic DNS Server Configuration (Testing - BIND Commands)

- `rndc`

- `/etc/rndc.conf`
  - default configuration file for the `rndc` utility
    - generated by `rndc-confgen -r /dev/urandom > /etc/rndc.conf`
    - NOTE: Comments about the rndc key at the bottom of this configuration file need to be copied to `/etc/named.conf` and uncommented (and named restarted if already running)
- can be used to perform certain actions on the 'named' service (the BIND server)
  - `rndc-confgen` • creates a default configuration file for the rndc utility (will display to terminal by default, redirect to the file `/etc/rndc.conf` for persistence)
      - if the configuration file is generated, by sure to change group ownership of the file to 'named' and set permissions to 0640 to prevent reading of the secret key used to control the service
  - `reload` • reloads all configuration and zone files
  - `dumpdb` • creates a dump file of server cache and/or zone files for the server
    - `dumpdb -cache` • just dumps the cache
    - `dumpdb -zone` • just dumps the zone information
  - `flush` • flushes all information from the 'named' cache
  - `flushname [domain]` • flush just the cached information for the indicated domain
  - `status` • display current named status
  - `reconfig` • reload the `/etc/named.conf` file and new zone files (note: does not reload existing zone files, even if changed)
  - `stop` • stops the named service

- `dig`
  - provides a plethora of information when querying a DNS server
  - **Example** • `dig @10.0.1.100 mylabserver.com`
    - would provide all information for the mylabserver.com domain using the domain server at address 10.0.1.100
- `host`
  - provides simple translation of name to IP for the indicated domain
  - **Example** • `host user.mylabserver.com`
    - would provide the address of the user.mylabserver.com host as well as other info (like mail server, if available)

- uses the values as specified in the `/etc/resolv.conf` file unless you indicate an alternate nameserver
  - **Example** • `host user.mylabserver.com 8.8.8.8`
    - would use the Google Public DNS nameserver at 8.8.8.8 rather than any nameserver in `/etc/resolv.conf` under the nameserver directive

207.2 Create and Maintain DNS Zones (Configuring for Zones)

- `/var/named`
  - default directory for zone files
  - files here define the host to address translation for the domains the BIND server is responsible for
- `/var/named/named.ca`
  - root server listing (will generally never modify the values in this file)
- `/var/named/named.localhost`
  - usually used to define a single host
  - see 'Appendix A - Sample Configuration Files' for an example
- zone files
  - zone files must contain references in the `/etc/named.conf` file
  - **Example** • for the zone called 'mydomain.com', your entries would look like:

    -
      ```
      zone “mydomain.com” {
      type master;
      file “named.mydomain.com”;
      };
      ```
    - this is a 'forward' zone file (translates names to addresses), indicating the type 'master' means that the current host is where changes to the zone file will be made and the file is the named of the file containing that zone information for the indicated domain
    - 'reverse' zone files (optional) translate addresses to names (sometimes is required for mail services)
    - **Example** • for the zone called 'mydomain.com', your entries in `/etc/named.conf` for reverse zone would look like:

      -
        ```
        zone “0.1.10.in-addr.arpa” {
        type master;
        file “db.10.1.0”;
        ```

```
};
```

- the format of the definition “0.1.10.in-addr.arpa” is the reverse of the first three octets of the domain address, this helps to indicate that it is a reverse lookup zone and is a standard naming convention

- secondary (slave) DNS server
  - often set up as a backup of the master, also contains zone information for the indicated domain
  - changes made to `/etc/named.conf` for the zone above (as an example)

    -
      ```
      zone “mydomain.com” {
      type slave;
      file “named.mydomain.com”;
      masters { 10.1.0.100; };
      };
      ```
    - the ‘masters’ directive should point to the primary/master DNS server IP

207.2 Create and Maintain DNS Zones (Zone Files and Record Types)

- zone file setup
  - see ‘Appendix A - Sample Configuration Files’ for an example of the ‘fw.mydomain.com.db’ file in our configuration
  - see ‘Appendix A - Sample Configuration Files’ for an example of the ‘0.1.10.db’ file in our configuration
- record types
  - **SOA** • start of authority, defines the authoritative information about a zone, contains:
    - **Name server** • domain name of the master name server (in our example - named. mydomain.com)
    - **Email** • DNS admin email (note: can be anything, but substitute a ‘.’ dot character for ‘@’ in the address)
    - **Serial number** • a number that indicates whether a zone needs to be updated to a slave, anytime you make a change to a zone file, this serial number should be incremented by some value (if a slave is configured, it will then initiate a zone transfer to update the zone)
    - **Refresh** • determines the frequency a slave server queries the master to determine if updates are needed
    - **Retry** • how long a slave will wait to retry a master query for an update
    - **Expiry** • when the slave server stops responding to DNS query requests if the master

continue to be unavailable

- **Minimum** • length of time to cache 'negative' responses (response that a domain or record does not exist before requerying for the value)

- address type
    - defines a direct 'name to address' translation
        - **Example** • `prod.mydomain.com    IN    A    10.1.0.101`
            - would define the 'prod.mydomain.com' server domain to the address of '10.1.0.101' if queried

```
+ can be given relative to the current domain or as a FQDN
```

- canonical name
    - allows you to define a host with more that one name (or role) in your domain
        - **Example** •

```
prod.mydomain.com    IN    A    10.1.0.101
logs                 IN    CNAME    prod
```

            - this would indicate that the name "logs.mydomain.com" translates to the name "prod.mydomain.com" (which then translates to the IP 10.1.0.101)

- name server
    - as every domain can have one (or more) name servers, they are defined as a NS (name server) record
    - although the master name server is defined in the SOA record (see Appendix A for the full listing), that server must still contain a NS definition
        - **Example** • `@    IN    NS    named.mydomain.com.`
            - defines the NS (nameserver) as 'named.mydomain.com'
            - note that all nameservers should have an associated NS record

- mail exchange
    - sending email services (called MTA - mail transfer agents) have to be able to figure out which host will handle inboud email for the mydomain.com domain, this is done by creating one or more MX (mail exchange) records
        - **Example** •

```
@    IN    MX    10    prodmail.mydomain.com
@    IN    MX    20    bkupmail.mydomain.com
```

- these two records define two mail servers, where the number indicates the priority (order) they should be tried (they can be different, as in the case of a primary and backup, or the same, as in the case of a load balanced mail environment)

- pointer
  - used in reverse lookup zone files so that the IP can be translated into the name
    - **Example** • `101.0.1.10.in-addr.arpa.    IN    PTR    prod.mydomain.com.` (NOTE: this is a FULL reverse record, generally not needed because of the $ORIGIN variable setting)
    - **Example** • `101    IN    PTR    prod.mydomain.com.` (NOTE: this is the more common short method for reverse records)
      - note the IP for the host is given in the reverse order of the IP in a forward lookup
- other record types
  - there are other record types, but these are by far the most common and the only ones you have to know for the exam

207.2 Create and Maintain DNS Zones (Finalize Master DNS Server Configuration)

- `/etc/named.conf`
  - see Appendix A for a full listing of the sample configuration for our 'mydomain.com' example
  - `allow-query` • add networks that are allowed to query the DNS server (default from caching nameserver setup is localhost only, in our example, we would add '10.1.0.0/24' to allow the entire network to query the server)
  - add the forward and reverse zone file references
    - forward zone
      -
        ```
        zone "mydomain.com" IN {
        type master;
        file "fwd.mydomain.com.db";
        allow-update { none; };
        };
        ```
    - reverse zone
      -
        ```
        zone "0.1.10.in-addr.arpa" IN {
        type master;
        file "0.1.10.db";
        allow-update { none; };
        };
        ```

207.2 Create and Maintain DNS Zones (Create Forward and Reverse Zone Files and Testing the Configuration)

- `/var/named/fwd.domain.com.db`
  - the forward lookup file for our domain server
  - see Appendix A for a full listing of our example domain
- `/var/named/0.1.10.db`
  - the reverse lookup file for our domain server
  - see Appendix A for a full listing of our example domain
- `named-checkconf`
  - utility that will check `/etc/named.conf` for syntax/formatting errors
  - NOTE: if no errors are detected, then no output will be displayed, if error, will provide a line number and some information about what may be wrong
  - the utility is a 'smart' utility in that it knows which options are valid or invalid and will indicate such as encountered
- `named-checkzone`
  - will check the specified zone file for syntax/formatting errors
  - full syntax is `named-checkzone [domain] [zone file]`
    - errors reported are not always the easiest to ascertain (most common error is a missing '.' at the end of certain lines - see Appendix A for examples of lines needing a trailing '.' character at the end of a name)
- `service named restart`
  - sysvinit method of restarting name server
- `systemctl restart named`
  - systemd method of restarting name server
- `dig`
  - utility to check DNS (forward lookup), can specify a different nameserver other than the default defined in `/etc/resolv.conf`
  - **Example** • `dig prod.mydomain.com @localhost`

        - will query the nameserver 'localhost' for the name 'prod.mydomain.com', returning ALL DNS record information on it

- `nslookup`
    - similar to dig in that it can return forward zone lookups, but can also return reverse lookups, specifying the nameserver to use
        - **Example** • `nslookup 10.1.0.101 localhost`
            - will use the nameserver 'localhost' to return the name associated with IP '10.0.1.101' (if there is one)
    - NOTE: this command is being retired (deprecated) but is an objective on the LPIC-2 exam

207.3 Securing a DNS Server (CHROOT Jails)

- chroot jail
    - NOTE: for the LPIC-2 exam, you will need to know the older 'chroot' method, most modern Linux distributions (CentOS 6+, Debian 6+, Ubuntu 11.04+, use a special package for the named process (called bind-chroot), this is not part of the exam, you are expected to know this older method)
    - running a process or service that is normally a 'root' level process or service as a 'non-root' user (sometimes called running with a service account) can still sometimes present security concerns (such as when system level files have read writes associated with them)
    - the solution is to place this process in a 'jail,' wholly contained in a subdirectory where the process can only see the filesystem in the subdirectory it is in (i.e. the subdirectory appears to be the root directory of the filesystem to the process)
    - an alternative to chroot jails is the SELinux security system; however, that is not an exam objective but is covered in great detail in other Linux Academy Linux certification preparation courses
- creating the jail
    - decide the directories that will serve as the jailed filesystem outside of the 'chroot' root filesystem
        - typical directories used:
            - `/chroot` • the 'root' filesystem of the jail
            - `/chroot/named` • the primary service directory
            - `/chroot/named/dev` • device files needed by the service
            - `/chroot/named/etc` • the location of any configuration files needed by the service
            - `/chroot/named/var/named` • the primary location of zone files (forward and

reverse) for the service

- `/chroot/named/var/run` • the location where process data is stored (i.e. PID file)

- copying key files
  - for named, the following files should be copied:
    - `/etc/named.conf` copied to `/chroot/named/etc`
    - `/etc/localtime` copied to `/chroot/named/etc`
    - `/var/named/*` - all files copied to `/chroot/named/var/named`
  - permissions and devices
    - `chown named:named -R /chroot/named`
      - recursively sets user and group ownership to the 'named' user account and group
    - `mknod /chroot/named/dev/random c 1 8`
    - `mknod /chroot/named/dev/null c 1 3`
    - `chmod 666 /chroot/named/dev/*`
- configured the service to start in the jail
  - `/etc/sysconfig/named` • the service options file
    - add `-t /chroot/named` to the end of the file

207.3 Securing a DNS Server (BIND - Split Name Server Setup)

- split DNS server (a dual server configuration is what is tested on the exam)
  - the ability to provide both internal and external DNS queries from one or more servers
  - some servers on your network are available publicly (web servers for example), while others on your network are not (HR or accounting for example)
- see diagrams to illustrate
  - configuration implementation is the same as single DNS server set ups, one with the domain internally and all hosts belonging to it (both internal AND external hosts) while the other with the domain externally and hosts intended for public access ONLY (partial zone information)
  - NOTE: very often, the internal DNS server will use the external DNS server as a forwarding DNS service (in your `/etc/named.conf`), this is a security tactic to abstract the internal network addresses from the external DNS servers (or any others that may be listening)

207.3 Securing a DNS Server (DNS Security Tools - Discussion, Keys and Signing a Zone File)

- TSIG (transaction signatures)
    - both private and public signatures are used to verify that the DNS response is coming from the right/trusted source
    - can be used for zone transfers AND DNS queries/responses
    - implemented most commonly with DNSSEC (Domain Name System Security Extensions)
- `dnssec-keygen`
    - used to generate the public and private key
    - `-a [encryption type]` • the encryption type to be used during key generation
    - `-b [#]` • the size of the key, in bytes, between 1 and 4096, depending on the key type
    - `-n [nametype]` • the owner of the key type (ZONE, HOST or ENTITY)
    - `[domain]` • the domain the key is to be generated for
        - **Example** • `dnssec-keygen -a RSASHA256 -b 2048 -n ZONE -f KSK mydomain.com`
            - would generated a 'RSASHA256' 2048 bit key type for the ZONE called mydomain.com
        - NOTE: entropy (`/dev/random`) is used and can take some time, particularly on headless/virtual systems, refill entropy with the /dev/urandom device by issuing a `rngd -r /dev/urandom -o /dev/random -b` command
    - will leave you with two files
        - `Kmydomain.com.[values].key` • the public key file
        - `Kmydomain.com.[values].private` • the private key file
- `dnssec-signzone`
    - uses the private key (created above) to sign a zone file, which can then only be verified using the public key
    - create the zone file first, then sign it with the private key
    - `-o [domain]` • the domain of the zone file
    - `[zone file]` • the zone file to sign
    - `[private key]` • the private key file to sign the zone with

- **Example** • `dnssec-signzone -o mydomain.com fwd.mydomain.com.db Kmydomain.com.+001+010203`
    - would create a new zone file called 'fwd.mydomain.com.db.signed' using the private key file 'Kmydomain.com.+001+010203'
    - this file would then be used (with DNSSEC turned on in the `/etc/named.conf` file) as the zone file for the domain 'mydomain'

Topic 208 Web Services

208.1 Implementing a Web Server (Apache - Configuration File and Directives)

- primary configuration file either `apache2.conf` or `httpd.conf` depending on distribution and version
    - common directories include `/etc/apache`, `/etc/apache2`, `/etc/httpd`, `/etc/httpd/conf`
    - configuration directives are identical regardless of locations
    - see Appendix A for a sample httpd.conf file
- `httpd.conf/apache2.conf`
    - `ServerRoot "[directory]"` • defines the Apache server primary configuration directory
    - `Listen [IP:Port]` • binds the Apache service to the IP and port (IP is optional, omitted would be for all addresses)
    - `DocumentRoot "[directory]"` • the default system wide directory where content files are located (can be overridden in individual virtual hosts - covered later)
    - `LogLevel [level]` • the amount and type of messages that will be logged in the error_log
        - valid levels are alert, emerg, crit, warn, error, info, debug, and notice
- directives
    - sections within the configuration file that contain one or more values defined for use
- service
    - either httpd or apache2 (again, depending on the distribution)
- alternative server start
    - apache2ctl or apachectl (foreground or background)

- `/var/www/html`
  - the default location (without making any change) that apache will access to to display site content
- `/var/log/httpd`
  - default logging location for the Apache access and error logs (access_log and error_log)
  - names of files and locations can be overridden in vhost files or changed in the primary configuration file

208.1 Implementing a Web Server (Apache - Enabling Perl and PHP Scripting)

- Perl module
  - `apache-mod-perl` or `libapache2-mod-perl2`
  - `mod_perl.so` (the module name that needs to be installed in the `/etc/httpd/modules` directory)
- enabling the module
  - add the module in the primary configuration file
  - alternatively, create an entry for loading the module in the `/etc/httpd/conf.modules.d` directory
    - the numbered configuration file should contain the line `LoadModule perl_module modules/mod_perl.so`
    - NOTE: an include statement for loading configuration files needs to be added to the main configuration file
- directive within the httpd or apache2.conf file

  -
    ```
    <Directory /var/www/html/perl-cgi>
     AllowOverride All
     SetHandler perl-script
     PerlHandler ModPerl::Registry
     PerlOptions +ParseHeaders
     Options ExecCGI
     Order allow,deny
     Allow from all
    </Directory>
    ```
  - when a Perl script is called for, it will look in `/var/www/html/perl-cgi` (or whatever other directory you indicate)
  - reload/restart the Apache service

- PHP module(s)
    - same process as Perl
    - add the module in the primary configuration file
    - alternatively, create an entry for loading the module in the `/etc/httpd/conf.modules.d` directory
        - the numbered configuration file should contain the line `LoadModule php5_module modules/libphp5.so`
        - NOTE: an include statement for loading the configuration files needs to be added to the main configuration file
    - unlike Perl, the PHP files will exist in the same directory as other content files are
    - reload/restart the web server to test

208.1 Implementing a Web Server (Apache - Security Settings and User Authentication)

- creating various limits within your configuration, including:
    - `StartServers [#]` • by default, a single process is started owned by root, that does not handle web requests, this setting will determine how many Apache user owned processes are started to handle client requests
    - `MinSpareServers` • as client requests are assigned to the Apache processes (httpd) that will serve them, new server processes need to be started to serve new requests, this setting will determine the minimum number of processes that will always be available for incoming connections
    - `MaxSpareServers` • will allow the Apache process to kill processes that are not serving requests once they exceed this number
    - `MaxClients` • this is the maximum number or Apache servers that can be started (effectively limiting the number of client requests that can be handled at any one time)
    - `MaxRequestsPerChild` • generally a limit that is used to prevent or more gracefully handle Denial of Service (DoS) attacks
- `htpasswd`
    - allows you to secure specific directories and/or files on a site, requiring user authentication before granting access
    - `-c [path of passwd file to create] [name of user]` • creates a password file and prompts for a password for the indicated user
        - **Example** • `htpasswd -c /etc/httpd/passwdfile linuxacademy`

- will create a new password file called 'passwdfile' and prompt for a password for the 'linuxacademy' user

- primary (apache2.conf or httpd.conf) configuration for a secure directory

  -
    ```
    <Directory /var/www/html/secure_dir>
      AuthName "Secure folder"
      AuthType Basic
      AuthUserFile /etc/httpd/passwdfile
      Require valid-user
    </Directory>
    ```

  - can be overridden within vhost files specific to a site (covered later)

- `.htaccess`

  - alternative method of securing a directory

  - primary (apache2.conf or httpd.conf) configuration for this type of secure directory

    -
      ```
      <Directory /var/www/html/secured_2>
        AllowOverride AuthConfig
      </Directory>
      ```

  - file is added to the directory to be secured (in our example, `/var/www/html/secure_dir/.htaccess`)

  - contains the directives as follows:

    -
      ```
      AuthName "Secure folder"
      AuthType Basic
      AuthUserFile /etc/httpd/passwdfile
      Requrire valid-user
      ```

208.1 Implementing a Web Server (Apache - Virtual Hosts)

- vhosts

  - allows you to host more than one website on a single Apache host/instance/server

  - name-based virtual hosts - the most common method where all domains share a single IP and are differentiated by the site name

  - IP-based virtual hosts - each domain has a separate IP (either IP aliases or multiple network devices)

- IP-based virtual host configuration (within primary configuration file)

  -
    ```
    <VirtualHost www.mydomain.com>
     ServerAdmin admin@mailprod.mydomain.com
     DocumentRoot /var/www/html/mydomain
    ```

```
    ServerName www.mydomain.com
    ErrorLog /var/log/mydomain/error_log
  </VirtualHost>
```

-
  ```
  <VirtualHost www.extradomain.com>
    ServerAdmin admin@mailprod.extradomain.com
    DocumentRoot /var/www/html/extradomain
    ServerName www.extradomain.com
    ErrorLog /var/log/extradomain/error_log
  </VirtualHost>
  ```
- restart of the apache/httpd service to load the virtual hosts and make the sites available

- name-based virtual hosts (within primary configuration file)
    - both sites will resolve to the same IP address via DNS

    -
      ```
      NameVirtualHost 10.1.0.110
      <VirtualHost 10.1.0.100>
        ServerName www.mydomain.com
        DocumentRoot /var/www/html/mydomain
      </VirtualHost>
      <VirtualHost 10.1.0.100>
        ServerName www.extradomain.com
        DocumentRoot /var/www/html/extradomain
      </VirtualHost>
      ```
    - restart of the apache/httpd service to load the virtual hosts and make the sites available

208.2 Apache Configuration for HTTPS (SSL Key Generation)

- SSL Certificates
    - using public and private keys, encrypts the transactions that take place between a client and a web server
    - certificates can be 'signed' two ways

        1. **Self signed** • this is when the site/server that generates the certificate 'signs' it, there is no independent certificate authority involved and browsers will warn or block a client from reaching a site with a self signed certificate, these are often used in non-production/test environments (like our demo will be)

        2. **Signed by a certificate of authority** • a signature file is created and then transmitted to a certificate authority (Verisign, Go Daddy, etc), these authorities include intermediate certificates in most browsers and are trusted to verify that the certificate on the site represents a secure connection with a valid destination

- `mod_ssl` (package to install)
  - the primary method of implementing SSL certificates on the standard Apache server (and the method that you can expect to see on the exam)
  - loaded in the primary Apache configuration file with `LoadModule ssl_module modules/mod_ssl.so`
  - in the `/etc/httpd/conf.modules.d` directory, a numbered configuration file will be created (be sure that the configuration files are loaded in the apache configuration like the Perl/PHP modules earlier)
- default self signed certificate
  - is accessible after the module is installed and configured
- `openssl`
  - used to generate the key, certificate signing request and/or the self signed certificate on your system
  - `genrsa` • the key type (RSA) to generate, DSA is available, but RSA is the preference for security
  - `-des3` • the algorithm used that specifies the encryption type (triple DES)
  - `-out [keyname]` • the key that will be generated
  - **Example** • `openssl genrsa -des3 -out mydomain.key 2048`
    - will generate a RSA Triple DES encrypted 2048bit key called 'mydomain.key' in the current directory
  - `req -new` • will request a new certificate or certificate signing request
  - `-key [keyname]` • the key to use for generating the certificate signing request (CSR)
  - `-out [signing request file]` • the certificate signing request file
  - **Example** • `openssl req -new -key mydomain.key -out mydomain.csr`
    - will generate a new certificate signing request called 'mydomain.csr' using the key called 'mydomain.key'
- `openssl-perl`
  - used to self sign a certificate (creating a full path - key, signing request and certificate)
  - `/etc/pki/tls/misc/CA.pl` (NOTE: exact path to this script may be different for your distribution)
  - generate a new certificate authority (self)

        - create a working directory (i.e. `/root/httpd`)
        - `/etc/pki/tls/misc/CA.pl -newca`
            - a large number of questions will be asked about site, company, etc
            - for self-signed certificates, most are optional except the password for the key
    - generate a new signing request
        - `/etc/pki/tls/misc/CA.pl -newreq`
            - a large number of questions will be asked about site, company, etc
            - for self-signed certificates, most are optional except the password for the key
    - generate a new certificate from the previous commands
        - `/etc/pki/tls/misc/CA.pl -signreq`
            - a large number of questions will be asked about site, company, etc
            - for self signed certificates, most are optional except the password for the key
    - in the working directory (in our example, `/root/httpd`), will be three files
        - newcert.pem (the certificate itself)
        - newkey.pem (the private key)
        - newreq.pem (the signing request - not needed after the certificate is generated)

208.2 Apache Configuration for HTTPS (Adding SSL Certificates to Configuration and Key SSL Directives)

- create a directory for the key and certificate files
    - `/etc/httpd/ssl` (or `/etc/apache2/ssl`)
    - `/etc/ssl`
        - any of these are acceptable locations
- copy the `newkey.pem` to your domain name
    - **Example** • `cp /root/httpd/newkey.pem /etc/httpd/ssl/mydomain.com-privatekey.pem`
- copy the `newcert.pem` to your domain name
    - **Example** • `cp /root/httpd/newcert.pem /etc/httpd/ssl/mydomain.com-selfcert.pem`

- enable the key and certificate in Apache configuration (httpd.conf or apache2.conf)
    - add the following directives:
        - `SSLCertificateKeyFile /etc/httpd/ssl/mydomain.com-privatekey.pem`
        - `SSLCertificateFile /etc/httpd/ssl/mydomain.com-selfcert.pem`
    - restart Apache service
- additional SSL directives to be aware of for the exam
    - `SSLCertificateChainFile` • a file that can be used for a browser to discover the certificate chain of authenticity if not included as trusted in the browser itself
    - `SSLCACertificateFile` • a file that can be used to authenticate a client (rather than authenticating the server), it is a combination (concatenation) of all certificates of authority
    - `SSLCACertificatePath` • part of authenticating clients, in this case, you would place all files in a directory rather than a single file combining them all
    - `SSLProtocol` • specifies the protocol to use and the version, often used for backward compatibility (valid values are SSLv2, SSLv3, ALL, TLSv1, TLSv1.1 or TLSv1.2)
    - `SSLEngine` • used to turn SSL support for sites on or off as part of the virtual hosts (if used)
    - `SSLCipherSuite` • indicates the cipher used to create the keys (like RSA)
    - `ServerTokens` • specifies what information is returned to a client about the server connection (OS, version, server type)
    - `ServerSignature` • creates footers containing information that can be used for debugging (on/off)
    - `TraceEnable` • turns on SSL tracing for debugging, off by default

208.3 Implementing a Proxy Server (Squid - Installation and Default Configuration Walkthrough)

- types of proxy servers
    - **Forward proxy** • client proxy that can block access to certain public resources, abstracts a client connection as the requested service sees a request as though it comes from the proxy rather than the client requesting it, some data can also be cached (static data - icons, images, etc) as well as providing logs of client network activity/requests
    - **Reverse proxy** • server proxy that can provide load balancing, caching of static data, offloading of SSL based traffic, abstraction of the server environment as the client only see the proxy and can compress data for increased speed
    - **tunnel proxy** • acts as an intermediary between two network connections (not an LPIC-2

objective)

- common proxy applications
  - **nginx** • can function as a web server, load balancer and reverse proxy server
  - **Squid** • can function as both a forward proxy as well as a reverse proxy
- `squid` (package for every distribution)
  - `/etc/squid/squid.conf`
    - primary configuration file for squid
      - `cache_dir [type] [cache directory] [max cache file size] [first and second level subdirectories under cache]` • when configured as a forward proxy, this setting will use the directory and settings to cache data
      - `http_port [port]` • the port that the proxy will listen to for connections (default is 3128)
      - `auth_param` • allows client authentication (through LDAP for example)
      - `http_access` • indicates the ACL name that is permitted to use the proxy as a forward proxy
      - `acl` • allows the grouping of source and destination networks, ports, addresses and/or security settings and associate those settings with a name that can be referred to in the configuration (access control list)
- default configuration
  - see Appendix A for a sample `squid.conf` configuration file
  - both acl and http_access examples are listed in the configuration file
- acl (format)
  - `acl [name] [type] [data]`
    - `[name]` • user designated name to reference the ACL with
    - `[type]` • src (source), dst (destination), time (allow access during), Safe_ports (http ports allowed), SSL_ports (SSL ports allowed)
    - `[data]` • ports, IPs, networks or ranges of addresses that are allowed
  - definitions in use
    - `all` • indicates all systems
    - `manager` • squid cache management
    - `to_localhost` • destination IP address of the local machine, used if localhost is the

destination

- `localhost` • destination IP address of the local machine, used if localhost is the source

- `http_access`
    - `http_access [allow/deny] [data]`
        - `[allow/deny]` • allow or deny access to the indicated connection, ACL or all
        - `[data]` • name of the ACL or action to apply the allow/deny to

208.3 Implementing a Proxy Server (Squid - Testing with a Client)

- two servers, one client, one proxy server
- test on port 3128
- client configuration for lynx
    - `export http_proxy:http://[private ip of lab proxy server:3128]`

208.4 Implementing Nginx as a Web Server and Reverse Proxy (Nginx - Basic Web Server Configuration)

- `nginx`
    - default package for every distribution
- `/etc/nginx/nginx.conf`
    - primary configuration file for nginx web server (and later, proxy configuration)
    - basic web service configuration requires the following changes:
        - ‘server’ section
            - `listen 80 default_server;` (for IPv4) OR
              `listen [::]:80 default_server;` (for IPv6)
              `server_name [servername];`
              `root [/path/to/site];`
        - optionally, ‘error’ sections
            - create any pages referenced in these sections

208.4 Implementing Nginx as a Web Server and Reverse Proxy (Nginx - Basic Reverse Proxy Configuration)

- reverse proxy features for nginx include:
  - load balancing
  - acceleration
  - client authentication
  - bandwidth management
  - protocol support (TCP, POP3, SMTP, IMAP reverse proxies)
- nginx
  - default package for every distribution
- `/etc/nginx/nginx.conf`
  - primary configuration file for the nginx reverse proxy server (as well as web server from previous example)
- proxy configuration
  - replace entire 'server' section as follows

    -
      ```
      server {
         listen 80;
         location / {
         proxy_pass http://[IP of server]:[port]
         }
      }
      ```
  - save and restart nginx

209 - File Sharing

209.1 SAMBA Server Configuration (Server Installation and Share Configuration)

- packages to install (server)
  - samba, samba-client, samba-common
- documentation (SAMBA3 is testable)
  - `/usr/share/doc/samba-3.x.x/htmldocs`
- `/etc/samba/smb.conf`
  - simplified (no comments or documentation) configuration file example

- `/etc/samba/smb.conf.example`
  - more complex, fully documented with all options, configuration example
- broken into sections for various items:
  - `[global]` • global server options
    - `workgroup` • NETBIOS domain name, used mainly when communicating with Windows systems
    - `server string` • text description of the server, can be whatever you want
    - `security` • the type of security to use on the share(s)
      - `users` • samba user account security
      - `[domain name]` • domain controller security (Windows)
      - `ads` • Windows active directory security (username mapfile)
    - `passdb backend` • how samba account info is stored (default is 'tdbsam')
    - `load printers` • yes/no for sharing all CUPS printers available on the server
    - `cups options` • CUPS options for printers (rarely changes or added)
  - `[homes]` • allows users to provide sharing of the home directories automatically
    - `comment` • text description of the share
    - `browseable` • yes/no, determines if the home share is visible from client tools or utilities when displaying available samba shares remotely (no would keep the share from showing, but it is still available by name)
    - `writeable` • yes/no, determines read only or read/write access of the share (NOTE: filesystem permissions will still apply, granting write access to the share but being restricted by the filesystem permissions - set share on server to 'rwx' to rely only on this setting)
  - `[printers]` • determines if printers are available through the CUPS printing system
    - `comment` • text description of the share
    - `path` • spool directory of the CUPS system
    - `browseable` • yes/no, determines if the home share is visible from client tools or utilities when displaying available samba shares remotely (no would keep the share from showing, but it is still available by name)
    - `guest ok` • yes/no, permits printing access to the guest account (if defined, see below)

        - `writeable` • always set to no since this is not a filesystem
        - `printable` • always set to yes since this defines a print share
    - `[optional/custom share]` • shares defined by the administrator
        - `comment` • text description of the custom share
        - `path` • the path to the directory to share
        - `guest ok` • yes/no, permits the guest account access to the share (if defined, see below)
        - `writeable` • yes/no, determines read only or read/write access of the share (NOTE: filesystem permissions will still
- create custom directory
    - Example • `mkdir /myshare && chown 777 /myshare` (relying on the permissions set in the share section of the configuration)
- `testparm`
    - utility to test the `smb.conf` file and provided shares/options
- services to start
    - smb (or smbd)
    - nmb (or nmbd)
- `nmblookup [WORKGROUP]`
    - should display the server and workgroup name

209.1 SAMBA Server Configuration (Security and Account Management)

- `smbpasswd`
    - utility to add a user account and password to the samba user database
    - `-a [user]` • add the user after prompting for a password (NOTE: the user MUST have a system account or be mapped in the usermap file, see below)
- `username map`
    - file created to map system accounts and samba accounts (important for Windows users without system accounts)
    - defined in the [global] section in /etc/samba/smb.conf

- **Example** • `username map = /etc/samba/usermap`
  - entries are keypair in nature
    - **Example (local account mapping)** • `jsmith = user`
    - **Example (Windows account mapping)** • `jsmith@hotmail.com = user`
    - NOTE: the aforementioned 'guest account' is not defined in the usermap, but rather in the section that defines the share the guest account should have access to (i.e. in the [custom share] section, set 'guest account = guest' and then create the 'guest' account using 'smbpasswd -a guest' as normal)

- `smbclient`
  - utility to test samba user account access to a share
  - `-U [user]` • the user to connect as
  - `[//address/sharename]` • the address and share name of the intended share
    - password will be prompted for
  - will be placed at the `smb:\>` command prompt, `help` will display a list of valid commands (cd, ls, dir, etc, similar to FTP commands)
- `smbstatus`
  - shows the status of the server, its account connections and shares being accessed

209.1 SAMBA Server Configuration (Client Installation and Access)

- packages to install (server)
  - samba, samba-client, samba-common, cifs-utils
- `smbclient`
  - utility to test samba user account access to a share
  - `-U [user]` • the user to connect as
  - `[//address/sharename]` • the address and share name of the intended share
    - password will be prompted for
  - will be placed at the `smb:\>` command prompt, `help` will display a list of valid commands (cd, ls, dir, etc, similar to FTP commands)
- mounting a share

- **Example** • `mount -t cifs -o username=user //10.1.0.100/myshare /mnt/mysambashare`
    - would mount the samba filesystem at 10.1.0.100 called `myshare` locally on the `/mnt/sambashare` directory, prompting for the password for the 'user' account
- fstab example • `//10.1.0.100/myshare   /mnt/mysambashare   cifs   username=user,password=secret   0   0`
    - accomplishes the same process, only on boot
    - NOTE: bad security as the password is in plain view

- samba credentials file
    - can be created anywhere, often in `/etc/samba` directory, formatted in keypair
        - `username=user`

```
password=secret
```

- fstab example with credential file • `//10.1.0.100/myshare  /mnt/mysambashare  cifs  credentials=/etc/samba/creds.txt  0  0`

209.2 NFS Server Configuration (NFS3 Server Installation, Configuration and Testing)

- packages to install
    - nfs-utils, nfs-utils-lib, rpcbind
        - rpcbind provides portmap
- `/etc/exports`
    - file that provides information about the directory to be shared, what network/client addresses can access it and what options are available to each
    - general format:
        - `/path/to/directory  hostname(options) [(global options)]`
            - NOTE: space delimited list for hostname(options) can be provided
        - **Example** • `/nfsshare user1(rw) user2(rw) user3 10.1.0.0/24(rw) (ro)`
            - would provide an exported share called `/nfsshare` as read/write to the host user1 and user2, default options to user3 (notice the space between host and options), read/write access to any client on the 10.1.0.0/24 network and global read only to every other host
    - other options:

        - `rw` • read/write
        - `ro` • read only
        - `sync` • all file writes are immediate
        - `async` • file writes are made to memory and flushed to disk periodically (can increase chances of corruption)
        - `root_squash` • do not map the client root account to the server root account (done for security to prevent local root account from being a share root account)
        - `no_root_squash` • map the local and remote root account IDs so they have the same permissions
- mapping of User IDs (UID) and Group IDs (GID)
    - to preserve permissions (and not have UIDs conflicting or causing other users to have different user's file access), it is important to map the UID/GID of each account intended to use NFS shares so they are the same on both the server and the client system
    - typically the reason that NFS is not a public share system, used privately on private networks because of the management (there are methods of automating this synchronization, like OpenLDAP, covered later in the course)
    - **Example** • `useradd –u 1101 –m user1`
        - will create the user and group called 'user1' with a UID/GID of 1101 and create the home directory for that user
- services to start
    - nfs and rpcbind
- processes that you may see:
    - `rpc.statd` • recovery of files (in the event server is rebooted while clients are connected and accessing files)
    - `rpc.quotad` • works with filesystem quotas (if defined)
    - `rpc.mountd` • for client mount requests (NOTE: may just be called mountd)
    - `rpc.idmapd` • for NFS4 (or hybrid NFS3/4 systems)
    - `nfsd` • primary server process handling clients (can also see rpc.nfsd or nfsd4/rpc.nfsd4 for hybrid systems)
- `rpcbind` (`portmap`)
    - `portmap` informs client connections how to reach a resource (like rpc) port, literally maps the port

- `rpcinfo -p`
  - will show all services available for nfs/rpc
- `portmap`
  - uses 'TCP Wrappers' library so that access can be controlled
    - `/etc/hosts.allow` • grants access
    - `/etc/hosts.deny` • denies access
    - `DEFAULT (no rule match)` • grants access
  - generally used to restrict access to a small list of hosts while denying everything else (ALL)
    - hosts.allow takes precendence (whatever is in this file gets access, regardless of what is in hosts.deny)
- `exportfs`
  - server utility to show what shares are available
  - can also be used to export a new share not defined in the `/etc/exports` file (as long as service is running)
    - **Example** • `exportfs -o ro 10.1.0.110:/newshare`
      - will provide global read only access to the `/newshare` directory on the server 10.1.0.110 (same server as run on)
  - changes made to `/etc/exports` directly are active after a reboot OR on issuing `exportfs -a` to reload configuration
- `nfsstat`
  - shows information from client connections
  - `-m` • show client mounts and associated configuration
- `showmount`
  - similar to nfsstat, but less info
  - `-a` • show all mount points

209.2 NFS Server Configuration (NFS3 Client Configuration and Share Mounting)

- packages to install
  - nfs-utils, nfs-utils-lib, rpcbind

- create a local mount point
    - **Example** • `mkdir /mnt/nfsshare`
- local and remote mapped users
    - be sure to set passwords for them
- NOTE: for the demo, be sure you have created a local host entry for the remote client name and private IP in order to meet the configuration (we used private IP in the configuration)
- mount the share
    - **Example** • `mount 10.1.0.110:/nfsshare /mnt/nfsshare`
- verify the mount

    -
      ```
      df -h
      mount
      cat /proc/mounts | grep nfs
      ```
- `/etc/fstab`
    - entry for persistence
    - **Example** • `10.1.0.110:/nfsshare  /mnt/nfsshare  nfs  defaults  0 0`
        - will mount the 'nfsshare' from server 10.1.0.110 on the local directory /mnt/nfsshare as an nfs filesystem with default access rights
    - additional options available for `/etc/fstab` (other than 'defaults')
        - `soft` or `hard` • if unavailable, soft will stop trying to mount, hard will continue trying (see timeo option)
        - `fg` or `bg` • determines if the attempt to mount happens in the foreground (waits for mount to succeed/fail) or background (boot can continue while silently attempting in the background - used with 'hard' above to prevent boot hanging)
        - `timeo=[#]` • timeout value before mount attempt fails (in tenths of a second)
        - `retrans=[#]` • how many times the system will retry mounting a share
        - `rsize=[#]` • maximum read size request allowed of the remote server (default 8192 or 8mb)
        - `wsize=[#]` • maximum write size request allowed to the remote server (default 8192 or 8mb)
        - `rw` or `ro` • attempts to mount as read/write or read only, but ultimately determined as specified in server /etc/exports configuration

Topic 210 - Network Client Management

210.1 DHCP Configuration

- DHCP
    - Stands for Dynamic Host Configuration Protocol.
    - Specifically, it is a network protocol that provides the ability for a router (or in this case, a Linux server) to automatically assign an IP address to a client system from a pre-determined range of IP addresses.
    - A client comes online and sends out a broadcast over the network, which is intercepted by the DHCP, and then an address is provided from the pool of addresses (either the next available address OR an assigned address 'reserved' for that client under specific configurations).
- DHCP packages (outside of client functionality) is not typically installed by default in most distributions (outside of specific 'server spins' of some enterprise class distributions like Red Hat or CentOS).
    - Packages can be:
        - dhcp
        - dhcp-server
        - isc-dhcp-server (older)
        - dhcp3-server (LPIC-2 Exam Objective)
        - dhcp4-server
- Primary Configuration File
    - `/etc/dhcpd.conf`
        - or can be copied from `/usr/share/doc/dhcp*/dhcpd.conf.sample`
        - The configuration can be boiled down to the following directives for our exam purposes:
            - `ddns-update-style [style];`
            - `ignore client-updates;`
            - `subnet [network] netmask [mask] { [options and directives] }`
            - `host [name] { [static network information] }`
            - `lease [address] { [lease directives] }`
- DHCP Assignments – Sequence of Events

- PC or server boots, DHCP client software/daemon starts.
- Client broadcast (DHCPDISCOVER), attempts to find a DHCP server on the wire.
- Router/switch forwards the DHCPDISCOVER to the proper DHCP server (as configured).
- Server receives DHCPDISCOVER and, based on the configuration of available addresses, the client's hardware address and/or hostname and the configuration of the DHCP server software, determines the appropriate address to assign to the machine originating the request.
- The address is (temporarily) reserved for the client machine and the DHCP server sends the client a DHCPOFFER with the address information contained within.
- The client, upon receipt, will respond with a DHCPREQUEST, to inform the DHCP server it intends to use the address.
- The server responds with a final DHCPACK, confirming the client and assigning a lease on the provided address for the specified (configured) period of time.

- DHCP Relay Agent
  - A special agent that can listen on subnets that do not have a DHCP server for DHCP requests (or BOOTP requests) and then forward those on to a specific DHCP server on another network.
  - Command to Run Agent:
    - `dhcrelay [IP of DHCP Server to Forward to]`
- Configuration File – Directive Details

  -
    ```
    ddns-update-style [style];
    ignore client-updates;
    ```
    - Determines whether you want the DHCP server to attempt to update the requesting clients DNS server addresses. Valid values are:
      - `none` • no attempt to update DNS servers is made
      - `ad-hoc` • retired method of updating based on a script value
      - `interim` • C language based update of DNS, named as such as it was intended as a temporary replacement of ad-hoc
      - `standard` • newest method (DHCP v4), incorporates new standards for Dynamic DNS services
  - Key Directives
    - `option routers [ip(s)];`
    - `option subnet-mask [subnet of client];`
    - `domain-name [“domain”];`
    - `domain-name-servers [ip(s)];`

        - `range [beginip endip];`
        - `default-lease-time [seconds];`
        - `max-lease-time [seconds];`
        - `host [name] { [static network information] }`
            - `hardware ethernet [mac address of host];`
            - `fixed-address [ip to assign];`
            - `option host-name [“hostname”];`
- Logging
    - `/var/log/messages` (Red Hat/CentOS Distributions)
    - `/var/log/daemon.log` (Debian/Ubuntu Distributions)

210.2 PAM Authentication (Overview)

- PAM authentication
    - A method used by many Linux utilities to authenticate users.
    - The implementation of PAM allows you to control that authentication process by making changes to the associated PAM configuration file(s) without having to alter the tools or utilities directly (abstracting the security so it remains separated from the tools).
- Benefits
    - Password Control
        - Size and complexity of passwords
        - Historical password values (cannot use the same password twice)
    - Environment Control
        - Session variables
        - Processes that can be started
    - User Control
        - Geographical (IP/host-based) login limitations
        - Root account limitations (where root can login from)
- configuration structure (general files)
    - each file will contain one or more lines, each three columns across:
        - 1st Column (Type) – valid values are ’auth, account, session, password’

        - each one is evaluated based on the 2nd column.
    - 2nd Column (Control Value) – valid values are 'required, sufficient, requisite, required, optional, include'
        - Once the 2nd column is reached, the evaluation will either be 'successful' or 'unsuccessful' and the action in the 3rd column is applied
    - 3rd Column – the action of the line takes place here, there are a large number of valid actions identified by the *.so (shared object/library) or by another category included by key word
  - Special Value – include
    - This keyword indicates that you can include the rules in any other file (set of PAM values) already defined (common inclusion is 'system-auth')
- configuration example (account authentication)
  - `account      required          pam_nologin.so`
    - This line will determine if an account has been provided (it is required by the rule), if unsuccessful, it will deny login (determined by the `pam_nologin.so` module). If an account has been provided, it is successful and will move to the next line in the module for any further evaluation.

210.2 PAM Authentication (Modules - pam_unix.so, pam_cracklib.so, pam_limits.so and pam_listfile.so)

- `pam_unix.so`
  - modifies password limitations and rules
  - `/etc/pam.d/passwd`
    - see Appendix A for an example of this (and included) files
    - includes the 'system-auth' rules for passwords (substack)
  - `/etc/pam.d/system-auth`
    - see Appendix A for an example of this (and included) files
    - uses the `pam_unix.so` module with options to determine how passwords are managed (i.e. md5 to determine the encryption type)
    - **Example** • add `remember=3` at the end of the file to remember the last three user passwords and prevent their reuse
  - options that can be used:
    - `md5` • encrypt the password with md5 algorith

        - `sha256` • encrypt the password with sha256 algorithm
        - `remember=[#]` • remember previous passwords (up to ‘x’) and deny passwords that are the same as those remembered
        - `nullok` • allow root to provision accounts with no password
- `pam_cracklib.so`
    - modify how passwords are changed on the system (what users can choose as a password)
    - options that can be used:
        - `retry=[#]` • the number of times a password can be retried before system stops attempts
        - `minline=[#]` • the minimum characters a password can be
- `pam_limits.so`
    - used and modified within the `system-auth` configuration file
    - exception to the module configuration options rule, values are altered outside of the `/etc/pam.d` files
    - `/etc/security/limits.conf`
        - configuration file to apply various limits that can be applied more specifically (since any values modified in `/etc/pam.d/system-auth` would be applied to ALL configurations that include it - LOTS)
        - see Appendix A for an example of this file
        - general configuration on each line is `domain    type    item    value`
            - `domain` • user, group, wildcard
            - `type` • soft or hard
            - `item` • core, data, fsize, memlock, nofile, rss, stack, cpu, nproc, as, maxlogins, maxsyslogins, priority, locks, sigpending, msgqueue, nice, rtprio
            - `value` • the value assigned to the item when the limit applies
        - **Example** • `user    hard    maxlogins    3`
            - this would limit the ‘user’ account to three simultaneous logins to the server at any one time
- `pam_listfile.so`
    - scans file contents to determine whether they can be accessed or not
    - install FTP server (`vsftpd`)

- options that can be used:
  - `item=[username]` • match the indicated users
  - `sense=[accept/deny]` • accept or deny that user
  - `file=[/path/to/file]` • access to the indicated users file (user names are within this file)
  - `onerr=[succeed/fail]` • if error, succeed (so other modules will be evaluated)

210.2 PAM Authentication (Authentication Order - nsswitch.conf)

- `/etc/nsswitch.conf`
  - determines the order that files/services are used to perform authentication on your system
  - see Appendix A for an example of this file
- valid values for the authentication files/services in this file:
  - `nisplus` • se NIS+ v3
  - `nis` • use NIS v2 (YP or yellow pages)
  - `dns` • use DNS (Domain Name Services)
  - `files` • use local files (like /etc/hosts)
  - `db` • use the local database (.db) files
  - `compat` • NIS compatibility mode
  - `hesiod` • user lookups done with Hesiod
  - `[NOTFOUND=return]` • stop searching if not found so far when reached

210.3 Configuring an OpenLDAP Server (Overview)

- OpenLDAP
  - In short, OpenLDAP is a distributed directory service. It stores information associated with users that can be used to authenticate them for login and provide other information about those users (home directories, roles within the organization, etc).
  - OpenLDAP is most commonly used in Linux, but can be compared to Active Directory on Windows as a service provided hierarchical based user information much the same as OpenLDAP.
- definitions
  - Object

    - Sometimes referred to as a record or an entry, represents a single item in the directory. This object provides a description based on the structure of the schema.

- Schema
    - This is the structure that is built to define the characteristics (or attributes) of an object. It also defines what can be stored in each attribute.
- Attribute
    - This is part of an object. One or more attributes make up an object, as defined by the schema.
- LDIF
    - Stands for LDAP Interchange Format. It is used to create objects within the OpenLDAP directory. These values are placed into a file and can be loaded into the directory with the `slapadd` command.
- DN
    - Stands for Distinguished Name. Each object in your directory has to have a unique name in order to provide structure. It is build with a CN and one or more DC (example – `cn=user,dc=linuxacademy,dc=com)`
- CN
    - Stands for Common Name and is the name of the object (often a username, but not always).
- SSSD
    - Stands for System Security Services Daemon. This provides authentication of user accounts for OpenLDAP (you want to use a different authentication service – i.e. who authenticates the authenticator?)
    - NOTE: Configuration of this items is NOT an exam objective, only that you know what it is.

210.3 Configuring an OpenLDAP Server (Installation and Initial Configuration)

- NOTE: practice with version OpenLDAP 2.0 to 2.3 for the LPIC-2 exam (recommend CentOS 5 as in demo)
- `openldap`, `openldap-servers`, `openldap-clients`
    - client and server packages necessary for OpenLDAP
- `/etc/openldap`
    - primary server directory

- `/etc/openldap/ldap.conf`
  - basic server configuration parameters
- `/etc/openldap/slapd.conf`
  - imports schema that are needed to define standard LDAP structures
  - see Appendix A for an example of this file
- edits that need to be done for our schema
  - `suffix` • change to `dc=mydomain,dc=com` (primary top level structure)
  - `rootdn` • change to `cn=root,dc=mydomain,dc=com` (admin account)
  - `rootpwd` • admin account password plain text value OR encrypted value (preferred, see `slappasswd` below)
  - `loglevel` • (1 through 32768, doubling) for example - 32768 only logs messages that get logged no matter what the loglevel setting is (more than one, list of numbers, space delimited)
- `slappasswd`
  - generates the encrypted value for the `/etc/openldap/slapd.conf` file `rootpw` value
- `/var/lib/ldap`
  - directory defined in `/etc/openldap/slapd.conf` as default ldap storage location must exist (mode 700) or slapd service will fail to start
- `slaptest`
  - allows the testing of the `/etc/openldap/slapd.conf` file
  - `-u` • enable dry run
  - `-v` • verbose messages
  - `-f [file]` • specify an alternative `slapd.conf` file (for testing)
- process start
  - `service slapd start` (may be named ldap - check `/etc/init.d`)
  - `systemctl start slapd`
- `ldapsearch`
  - client utility, allows the searching of the directory, in this context, testing our DN
  - `ldapsearch -x -b ‘’ -s base ‘(objectclass=*)’ namingContexts`
    - `-x` • simple authentication to connect

- `-b [searchbase]` • starts search at indicated level (`""`, for example, starts at top level)
- `-s base [type of object class=value] [context]` • the return order and values
- see `ldapsearch` additional command details in next section

210.3 Configuring an OpenLDAP Server (LDIF Creation for Adding Objects)

- creating entries for objects
  - LDIF file
    - using a small example for the domain "domain.com", a basic LDIF file consists of the following:

      -
        ```
        dn: dc=my-domain,dc=com
        dc: my-domain
        description: our test LDIF
        objectClass: dcObject
        objectClass: organization
        o: mydomain example
        ```
- schema
  - organizes how data (objects and attributes) are stored in your directory
  - `objectClass: dbObject` • refers to attributes for ALL objects added
  - `objectClass: [class]` • refers to an object class and its definition in one of the included schema in the `/etc/openldap/slapd.conf` file
    - each object definition has a Object ID associated with it (globally unique and assigned/ maintained by Internet Assigned Numbers Authority - IANA)
    - description as to what it is used for
    - list of attributes that are required (MUST) and which are allowed but not required (MAY)
      - see `/etc/openldap/schema/core.schema` for an example of "organization" objectClass definition
- schema white page
  - a schema that provides information about accounts/users (telephone company book reference)
- `ldapadd`
  - allows you to add your LDIF file definitions/objects to the directory server

    - `-x` • simple authentication (although rootdn password must be given)
    - `-D [admin account DN]` • the admin account set up in `/etc/openldap/slapd.conf`
    - `-W` • the `ldapadd` command query for the supplied rootdn account password
    - `-f [ldif file]` • the name of the LDIF file to import
        - **Example** • `ldapadd -x -D “cn=Manager,dc=my-domain,dc=com” -W -f myfile.ldif`
            - will take the records defined in the LDIF file called ‘myfile.ldif’ and add them to the directory
- `slapcat`
    - show the changes made
    - output is in LDIF
        - HINT: copy/paste this output as a template for creation of additional objects as needed
- need to be aware of
    - `ldapmodify` • uses an LDIF file to modify an EXISTING record/object
    - `ldapdelete` • deletes a record/object specified (NOTE: no LDIF file is needed)
    - `slapindex` • optimizes existing records (by creating an index) for reading data more quickly
- access control
    - determines who can do what on the directory
    - complex implementation
    - NOTE: out of scope for the exam

210.4 LDAP Client Usage (Client Utilities for Searching, Adding and Deleting Records)

- `ldapadd`
    - covered in server configuration
    - `-x` • simple authentication (although rootdn password must be given)
    - `-D [admin account DN]` • the admin account set up in `/etc/openldap/slapd.conf`
    - `-W` • the `ldapadd` command query for the supplied rootdn account password
    - `-f [ldif file]` • the name of the LDIF file to import

- `ldapsearch`
  - allows you to specify records to search for in the directory
  - `-x` • simple authentication
  - `-b [base]` • the 'base' to search from
  - `[object class]` • the objectClass filter
  - `[field]` • the field to search (applied to the filter)
  - **Example** • `ldapsearch -x -b 'ou=people,dc=my-domain,dc=com' '(objectclass=*)' *`
    - would search and display all records in the 'people' OU (organizational unit) in the dc my-domain and com, for all objectclasses and fields within them
  - **Example** • `ldapsearch -x -b 'ou=people,dc=my-domain,dc=com' '(cn=Bruce Wayne)' uid`
    - would search and display the record in the 'people' OU in the dc my-domain and com for the CN of 'Bruce Wayne' and then display the UID field
  - `-L` • display LDAP v1 formatted output
  - `-LL` • show results and version only
  - `-LLL` • results only displayed
- `ldappasswd`
  - used to change USER passwords
  - `-x` • simple authentication
  - `-D [base]` • the base that the admin/root record exists in that has authorization to make changes (i.e. Manager or root record as defined in your `/etc/openldap/slapd.conf` file)
  - `-s [new password]` • plain text new password value
  - `-W` • prompts for rootdb password
  - `[uid/CN record and base]` • what record to change
    - **Example** • `ldappasswd -x -D "cn=Manager,dc=my-domain,dc=com" -s NewPwdValue -W 'cn=Bruce Wayne,ou=people,dc=my-domain,dc=com'`
- `ldapdelete`
  - allows the deletion of the indicated record/object
  - `-x` • simple authentication
  - `-D [base]` • the base that the admin/root record exists in that has authorization to make changes (i.e. Manager or root record as defined in your `/etc/openldap/slapd.conf` file)

- `-s [new password]` • plain text new password value
- `-W` • prompts for rootdb password
- `[uid/CN record and base]` • what record to change
- **Example** • `ldapdelete "uid=bwayne,ou=people,dc=my-domain,dc=com" -x -D "cn=Manager,dc=my-domain,dc=com" -W`
    - would delete the UID (or CN if indicated) 'bwayne' from our director

Topic 211 - E-Mail Services

211.1 Using E-Mail Services (E-Mail Overview)

- SMTP
    - Simple Mail Transfer Protocol. It is a protocol that really defines how e-mail is transferred and saved and is part of the TCP/IP application layer as well as settings rules that e-mail applications follow.
- MUA
    - Mail User Agent. This is whatever application you use to create and send e-mail (Thunderbolt, Evolution, SquirrelMail, etc).
- MSA
    - Mail Submission Agent. Acts as an intermediary or gateway between the MUA and an MTA to start the transfer of e-mail.
- MTA
    - Mail Transfer Agent. Accepts e-mail from the MUA and sends it (if needed) to the receiving mail server (another MTA if this is not the destination). There are a number of MTA server in Linux (Postfix, which we will use, sendmail and more).
- MDA
    - Mail Delivery Agent. Receives e-mail from the MTA and then delivers it to the local mail spool for retrieval by any of dozens of client e-mail applications. Sometimes an MTA can also function as an MDA, but often (procmail for example), they are independent applications that can also filter mail (like spam).
- POP
    - Post Office Protocol. Used by MUAs to get e-mail (covered in more detail later).
- IMAP

    - Internet Message Access Protocol. Used by MUAs to get e-mail (covered in more detail later).
- MX Record
    - These are the mail DNS records we created earlier in this course. These records are used by MTAs to determine the authoritative mail server for any particular e-mail message.

211.1 Using E-Mail Services (Postfix Key Configuration and Input Files)

- `sendmail`
    - older SMTP MTA, complex configuration (awareness of for exam)
- `exim`
    - newer than sendmail as an MTA, less complex configuration (awareness of for exam)
- `postfix`
    - a mail transfer agent (server) that we will configure to send and receive e-mail
- `/var/spool/postfix`
    - where postfix will store e-mail messages (within various subdirectories)
- `/etc/postfix/main.cf`
    - primary configuration file for the postfix server
    - settings are typically key/value pairs (something=value)
- `postconf`
    - utility that can be used to make postfix configuration settings rather than directly by editing the configuration file
    - `-n` • only display custom settings within the main.cf file
    - `[specific setting]` • will display JUST the indicated setting (if exists)
    - changing the value simply requires a new value
    - **Example** • `postconf command_directory=/sbin`
        - will set the `command_directory` value to `/sbin` (default is usually `/usr/sbin`)
- key settings
    - `myhostname` • FQDN of the system (server postfix is running on), should be detected by postfix by default
    - `disable_vrfy_command` • typically set to 'yes' on public servers in order to prevent e-mail

address mining

- `mydomain` • domain of the hostname
- `myorigin` • when a client does not indicate its hostname, this value will be used, if nothing is set, the hostname is used
- `inet_interfaces` • which available interfaces to bind/listen on, by default is localhost, on public servers, can be set to 'all' (leaving at localhost means that it only accepts e-mail from the local system)
- `mydestination` • set to myhostname by default, this is a list of all the domains (or systems/hosts) that this postfix server will accept e-mail for
- `relay_domains` • a comma delimited list of one or more domains that postfix can function as a relay for (if desired)
- `relayhost` • if messages are being forwarded to an outbound SMTP server, this will be the destination of the server

- `/etc/aliases` (depending on version, can also be `/etc/mail/aliases`, mainly sendmail systems)
    - a list of e-mail aliases and their intended destination addresses used by postfix for routing e-mail
    - certain aliases will always be present
        - **Example** • `mailer-daemon: postmaster`
    - general format is `alias: local_acct`
        - **Example** • `clark.kent: ckent`
    - more than one destination can be an alias
        - **Example** • `justiceleague: ckent,bwayne,ballen,dprince,jonnjonnz`
- `newaliases`
    - command line utility for taking new alias entries and adding them/creating them in the w`/etc/aliases.db` file used by postfix
- `/etc/postfix/virtual`
    - similar to aliases, but used to redirect e-mail to virtual destinations (common in multiple domain e-mail implementations)
    - format is similar to aliases
        - **Example** •

```
ckent@justiceleague.com  superman
ckent@thedailyplanet.com  clarkkent
```

- if you are using the virtual file, edit the `virtual_alias_maps` setting in the `/etc/postfix/main.cf` file and set it to `hash:/etc/postfix/virtual`
- this file must then be converted to binary using the 'postmap' utility
    - **Example** • `postmap /etc/postfix/virtual`
    - restarting postfix will then allow the configured virtual settings to take effect

- `mailx`
    - command line utility to quickly test postfix
    - `-s [text]` • subject of email
    - `[address]` • local or remote address to send e-mail
    - confirmed by sending test message one, viewing `/var/spool/mail/[username]`
    - stopping postfix, sending test message two, viewing same
    - restarting postfix, viewing same (now second message appears)
- `/var/log/maillog`
    - default mail logging location, all email transactions and service information contained within

211.2 Managing E-Mail Delivery (Rules Based Message Management)

- `procmail`
    - may be installed by default, but if not, procmail package is available for every distribution type and version
    - can apply filters for spam, automatic backups of user emails, filters based on address, etc
    - awareness of what it is, configuration for the LPIC-2 exam is out of scope
- sieve
    - language for complex filtering and sorting rules that can be applied to headers, size, senders and recipients

211.3 Managing Remote E-Mail Delivery (Dovecot Installation and Configuration)

- courier server
    - can provide IMAP and POP functionality (courier-pop and courier-imap packages)

    - combination of SMTP, POP, IMAP, LDAP, SSL and HTTP all in one (MTA, MDA, directory and web)
    - configuration and set up is outside the scope of the LPIC-2 exam, awareness of what it is is all we need to know
- dovecot
    - supports the IMAP and POP protocols as well as both major mailbox types (Maildir and mbox)
    - generally, on modern distributions, the default MDA
- `/etc/dovecot/dovecot.conf`
    - the primary configuration file for the server
- `dovecot`
    - utility to view the configuration
    - `-n` • display just the configuration items (minus all the commentary)
    - format of the settings are standard key/pair (something=value)
- key settings
    - `protocols` • which protocols to support (imap, pop3 or both)
    - `listen` • which interface dovecot will bind to ('*' equals all IPv4 interfaces)
    - `base_dir` • directory that dovecot data is located in
    - `!include` • allows the inclusion of external configuration files
        - **Example** • `!include conf.d/*.conf`
            - will include all configuration files in the relative path conf.d/ directory
    - `mail_location` • allows you to define custom mailbox locations
        - **Example** • `mail_location = mbox:~/maildir`
            - indicated that the 'mbox' mailbox is in each user's home directory, in a subdirectory called 'maildir'
            - special characters can be used:
                - `%u` • username
                - `%d` • domain
                - `%n` • user (of @domain)
                - `%h` • user's home directory

- `/etc/dovecot/conf.d`
  - external configuration files for dovecot, that are typically included in the configuration
  - NOTE: caution, changes made to parameters in any of these files that are duplicate settings from the dovecot.conf configuration file will be overridden by what is in dovecot.conf over these configuration files if after the !include statement (if before, the configuration setting in the external file will take precedent)
  - the numbers in front of some of the files indicate the order they are loaded/applied in the service
- testing
  - dovecot will listen on port 110
  - test with telnet
    - **Example** • `telnet localhost 110`
- `doveconf`
  - utility to dump the active configuration in more 'human readable' format
  - `-a` • show all settings and their current values
  - `-c [configuration file]` • read from the indicated configuration file (`/etc/dovecot/dovecot.conf` by default)
  - `-f [filter]` • display only the indicated value for the filtered condition
  - `-m [module]` • only show settings for the indicated module (protocols - imap, pop3, lmtp)
  - `-n` • show only non-default value settings
- `doveadm`
  - dovecot administration tool
  - `reload` • reload the configuration
  - `stop` • stops the service and all processes
  - `-f [formatter]` • formatter of output (values are flow for key/value, pager for key: value on own line, tab for table header with tab separated lines, table for table header with adjust value lines)
- certificates/SSL/TLS configuration
  - within the `conf.d` directory
  - `10-ssl.conf` file contains references to the certificates
  - generate certificate and key with OpenSSL as demonstrated in Apache portion of course

- use the default self-signed certificates included in the package
- test the certificates:
  - `openssl s_client –connect [email server]:pop3s (or imaps)`
    - valid certificate will show certificate info, self signed will connect but will show an error as a result of self-signed certificate

Topic 212 - System Security

212.1 Configuring a Router

- Requirement – a Linux system with at least TWO network interfaces on two separate networks.
- IP Forwarding
  - Kernel Setting
    - `echo “1” > /proc/sys/net/ipv4/ip_forward`
  - `sysctl.conf`
    - Add the following line
    - `net.ipv4.ip_forward = 1`
- To Include forwarding of IPv6:
  - `echo “1” > /proc/sys/net/ipv6/conf/all/forwarding`
- Private (Non-Routable) Networks
  - Private network ranges have been set aside for use inside corporate networks. They are not able to communicate directly with internet hosts and require a firewall or similar device to ‘translate’ (NAT – Network Address Translation) the originating IP to a public IP that can forward the traffic on their behalf.
    - Ranges are:
      - 192.168.0.0 to 192.168.255.255
      - 10.0.0.0 to 10.255.255.255
      - 172.16.0.0 to 172.31.255.255
- IPTABLES – Key Terms
  - Responsible for multiple tasks:
    - Restrict incoming/outgoing network traffic (packets) to/from the local machine

- Forward network traffic (packets) for the local machine
- Restrict network traffic (packets) for the local machine
- Perform Network Address Translation (NAT)
- Modify (mangle) network traffic (packets), including change the packet headers (if needed based on configured rules)

- Rules are configured and applied based upon certain conditions that each packet meets within one or more defined zones in the firewall.

- IPTABLES – Key Terms and Paths
  - Filtering Points
    - How rules are broken down within the firewall system.
    - Each point, you can create rules (sometimes referred to as a chain) that can apply to the packet passing through.
  - Inbound Packet Path

1. **PREROUTING** • configured to block, redirect or allow the packet to the next filtering point. Commonly, used to redirect the packet to another address and/or port (DNAT – Destination NAT).
2. **Destination Point** • If destination is local to the firewall's address (this machine), sent to the INPUT filtering point. If bound for another network, sent to the FORWARD filtering point.
3. **INPUT** • configured to be blocked, logged or sent to the local system to be handled by the appropriate client, application or service.
4. **FORWARD** • configured to blocked, logged or sent to the POSTROUTING filter point.
5. **POSTROUTING** • make changes to the packet as it exits the firewall, commonly used to do masquerading.
6. **OUTPUT** • packet is sent from the firewall out to the network to its final destination (NOTE: rules are not generally applied at this filter point).
   - NOTE: at each point, there are different rules (chains) and those rules are applied in the order they appear in the chain.

- IPTABLES – Tables within Filter Points
  - Tables that can be present in filter points are called FILTER, NAT, MANGLE.
  - Not every filter point has all three tables represented (see below)
    - PREROUTING (NAT, MANGLE)
    - INPUT (FILTER, MANGLE)

        - FORWARD (FILTER, MANGLE)
        - OUTPUT (FILTER, NAT, MANGLE)
        - POSTROUTING (NAT, MANGLE)
    - As a result of the tables in each filter point, you can define rules (chains). For example, at the filter point of FORWARD a set of rules designed to MANGLE packets would be the FORWARD-MANGLE chain.
- IPTABLES – Definitions
    - Rule
        - Something that is applied to a network packet (an action). The result of which is boolean (true or false).
            - True result – a target tells the firewall what to do with the packet next, additionally, no further rules are evaluated and the ruleset (chain) will exit.
            - False result – the packet moves to the next rule in the chain or the next chain to be evaluated further.
            - NOTE: this is an important concept since the order of rules in a chain can have a large impact on how the packet is affected by them, getting the order correct is the most important step.
- Rule Examples
    - Block all network traffic client to server
        - `iptables –t filter –A INPUT –s 10.1.0.100 – DROP`
        - Options
            - `-t filter` (applies to filter point)
            - `-A INPUT` (applies to INPUT table)
            - `-s 10.1.0.100` (source client IP)
            - `-DROP` (action to apply)
    - List all rules for filter-chain
        - `iptables –t filter –L INPUT`
        - Sample

            -
              ```
              Chain INPUT (policy ACCEPT)
               target  prot    opt source     dest
               DROP    all     --  10.1.0.100 any
              ```

- Block packets from interface
  - `iptables -t filter -A INPUT -i enp0s3 -j DROP (or REJECT)`
- Block packets in a port range
  - `iptables -t filter -A INPUT –m tcp -p tcp -dport 1023:2047 –j DROP (or REJECT)`
- Now we have three rules in our chain to evaluate each packet until matched (although because of the first rule - block everything - these other two rules would never be evaluated).

- IPTABLES – Options
  - Options outside of adding rules to a chain:
    - `–D` • allows you to delete rules
    - `–F` (or `--flush`) • flushes all (or indicated chain) of rules
    - `–P` • changes the default policy for the chain (can be set to DROP or ACCEPT)
    - `–v` • typically used with `–F` (or `--flush`) to provide additional output
  - Saving IPTABLES Rules (for reload or restore on reboot):
    - Rules stored in `/etc/sysconfig/iptables`
    - Created with command `iptables–save > /etc/sysconfig/iptables`
    - Restored from this file with command `iptables–restore < /etc/sysconfig/iptables`
- IPTABLES – Advanced Options
  - Your firewall can be used to forward a connection port to another server and port. For example:
    - `iptables -t nat -A PREROUTING -p tcp -dport 80 -j DNAT --to-destination 10.1.0.220:80`
      - In this case, IPTABLES is using the NAT-PREROUTING chain to check an incoming TCP packet on port 80. It is then sent (forwarded) to IP 10.1.0.220:80.
      - The `–j DNAT` option tells your firewall to use DNAT (Destination NAT) so that the originating host is not aware of the final address of the packet, which is only known to the firewall. IPTABLES maintains responsibility for ‘remembering’ the redirect and handling it.
  - Remembering our private networks, in general, setting up NAT capability on IPTABLES requires the line:
    - `iptables -t nat -A POSTROUTING -o enp0s1 -j MASQUERADE`
      - NOTE: the interface will depend on the IPTABLES system interface name and

configuration and may differ from the example (but it needs to be the internal network/ private interface).

212.2 Securing FTP Servers (Server - vsftpd)

- `vsftpd` (name of package to install and daemon)
    - 'very secure FTP daemon'
    - default for most modern distributions
    - although considered 'safer' than standard FTP because of the way the process uses the underlying filesystem, it is still not an encrypted transaction/community from client to server
- `/etc/vsftpd/vsftpd.conf`
    - primary configuration file for the vsftpd service
    - key settings (anonymous related)
        - `anonymous_enable` • yes/no, not a secure setting (NOTE: default value is YES)
        - `anon_upload_enable` • yes/no, allows the anonymous user to upload content (default is NO)
        - `anon_mkdir_write_enable` • yes/no, allow anonymous to create new directories (default is NO)
        - `anon_max_rate` • rate limit (in bytes) per second that an anonymous user can transfer files
        - `local_enable` • yes/no, disables local user accounts from logging in, common if anonymous access is allow (default is NO)
        - `dirmessage_enable` • yes/no, display contents of the .message file when entering a directory (default is NO)
    - other key settings
        - `userlist_enable` • yes/no, allows you to define a file that will allow or block users access to the server (default is NO)
        - `userlist_file` • if `userlist_enable` is set to YES, this defines the file that contains the list of user accounts (default /etc/vsftpd/user_list)
        - `userlist_deny` • yes/no, specifies whether the users in the 'user_list' file are allowed or denied access, if YES, the users in the account are DENIED access, if NO, ONLY the users in the file can access the server
            - note this is an alternative method to the PAM implementation that gives the flexibility to allow only a few people access in a big organization (so in an organization of 1000 where only 3 should have access, it is not necessary to add 997 to the PAM list for denying

access)

    - `banner_file` • allows the setting of a text file with a custom message to display, default if not set is the vsftpd version information
    - `ftpd_banner` • text (not a file) that can be displayed, short message under 64 characters
    - `chroot_local_users` • yes/no, will place local user accounts in a chroot jail so they can access ONLY their home directory
    - `write_enable` • yes/no, sets write permissions for regular users
    - `max_clients` • limit number of FTP client connections at one time
    - `max_per_ip` • limit how many FTP clients per IP can connect at one time
    - `local_max_rate` • the rate limit (in bytes) per second that local users can transfer files

- `/etc/vsftpd.ftpusers`
  - adding a user account to this file prevents their access regardless of whether 'local_enable' is set to YES (NOTE: exists to support /etc/pam.d/vsftpd file as demonstrated earlier in the PAM section of the course)
  - root is almost always a member of this file

212.2 Securing FTP Servers (Server - Pure-FTPD, ProFTPD and Active vs. Passive Connections)

- available more commonly for Debian based distributions (CentOS 7 exception)
- `pure-ftpd`
  - another, small, secure (ish) FTP server
- no configuration file is required, all configuration is done when launching the service at the command line
  - `-S [IP,PORT]` • the address and port for the server to listen on
  - `-c [#]` • limits concurrent connections
  - `-C [#]` • max number of connections per IP
  - `-l [#]` • max idle time (seconds) before auto disconnect
  - `-e` • permit anonymous access
  - `-E` • disable anonymous access
  - `-B` • start in the background (daemon mode)

- `proftpd`
    - another ftp server (configuration and setup outside the LPIC-2 objectives)
    - offers 'Apache-like' configuration file (`/etc/proftpd.conf`)
        - configuration similar to sections
            - `<Section> directive value </Section>`
- active connections
    - by default, ftp uses 'active' connections, your host will use a 'random' unprivileged port (>1023) to connect to the server on port 21
    - your host communicates and transmits login credentials, etc
    - once transfer begins, those transactions take place on server port 20 and client ports >1023
    - all commands from client to server (and responses back) are sent on the original ports (established when connected) while file transfers use the new port (20) while the client picks a new random port >1023 each time
- passive connections
    - the server does not start data transfer communication, it waits for the host to establish a connection (and as a result, performance is slightly better for the client and much easier on the firewall)
- active connections are default, passive connections are initiated by using the 'passive' command once logged into the server

212.3 Secure Shell (SSH Configuration)

- `openssh-server` (package on some distributions)
- `sshd` (package on some distributions)
    - `/etc/ssh`
        - directory for all SSH configuration file
    - `/etc/ssh/sshd_config`
        - openssh-server (sshd) primary configuration file
        - key settings
            - `Protocol [#]` • 1, 2 or both 1,2 (protocol 1 is less secure)
            - `ListenAddress [IP[:IP]]` • one or more (colon delimited) IP addresses for the server to listen on for connections

- `Port [#]` • port for the server to listen on (default is 22)
- `LogLevel [value]` • the level of logging that SSH will generate (valid values in order of verbosity are: QUIET, FATAL, ERROR, INFO, VERBOSE, DEBUG, DEBUG1, DEBUG2, DEBUG3)
    - logs to `/var/log/auth.log` (Debian-based systems)
    - logs to `/var/log/secure` (Red-Hat based systems)
- `PermitRootLogin [yes/no]` • allow root to login over SSH (force `su` for root access)
- `AllowUsers [user [user]]` • one or more user accounts (space delimited) that are allowed access
- `DenyUsers [user [user]]` • users who cannot connect (space delimited)
    - NOTE: these settings are one or the other, if both are defined, DenyUsers will take precedence
    - wildcards (? *) can be used for partial accounts (i.e. joh* will represent any name beginning with 'joh')
- `PasswordAuthentication [yes/no]` • users provide an account name and password for authentication
- `PubkeyAuthetication [yes/no]` • if yes, public keys can be used to access the system (ssh-keygen)
- `Banner [file]` • text file whose contents are displayed BEFORE user login
- `PrintMotd [yes/no]` • when set to yes, the `/etc/motd` contents will be display AFTER authentication
- `X11Forwarding [yes/no]` • allows X commands to be run with SSH clients (as long as they supply the `-X` or `-Y` option on connection)

`+ MaxAuthTries [#] • how many attempts at a password a user is allowed before disconnection (default is 6)`

- `PermitEmptyPassword [yes/no]` • set to no, denies access to any account that does not have a password

- `/etc/ssh/ssh_config`
    - ssh client utility configuration file (ssh, sftp, scp, etc)
    - `~/.ssh/config`
        - user directory SSH configuration file that overrides conflicting settings with the system client utility configuration file

        - many settings are related (or identical) to those found in the sshd_config file (see below)
    - NOTE: regardless of the existence of a home directory SSH configuration or a setting in the system SSH configuration file, command line options OVERRIDE those settings
        - order of precedence
            1. command line
            2. user SSH config file
            3. system SSH config file

212.3 Secure Shell (SSH Client Tools)

- `ssh`
    - client utility to initiate a connection to an openssh-server
    - `-l [username] [server name]` • allows you to log in as the named user (instead of the current user) to a remote system
    - `[user]@[server]` • alternative method of logging in as named user (instead of the current user) to a remote system
    - `[server name] [command(s)]` • allows the remote execution of the indicated command (with the right credentials)
- `scp`
    - copy utility that allows secure transfer of files to/from remote servers running openssh-server
    - **Example** • `scp /home/user/test.txt 10.1.0.100:/home/user`
        - will copy the local file 'test.txt' to the remote server '10.1.0.100' and place it in the '/home/user' directory on that server (NOTE: the user copying the file must have write access to the home directory)
    - **Example** • `scp user@10.1.0.100:/home/user/myfile.txt /home/user`
        - will copy the remote file, connecting as 'user' to '10.1.0.100' and copying the 'myfile.txt' to the local '/home/user' directory (NOTE: the user used to login to the remote system must have read access to the file and directory being copied)
    - `-r` • recursive copy
    - `-q` • quiet (no messages)
    - `-p` • keep all permissions and time/date stamps
    - `-v` • verbose messaging

- `sftp`
    - allows connections and transfers of files to a remote server running openssh-server, using commands similar to those used by ftp clients
    - **Example** • `sftp user@10.1.0.100`
        - will prompt for the account 'user' password for connection
        - using typical filesystem/ftp commands, you will have access to the same files/directories on the remote system as the account would if logged in via SSH
- `~/.ssh/known_hosts`
    - adds connection information (IP, hostname and associated RSA key fingerprint) for future connections
        - first attempt asks for confirmation to connect and add the information
        - subsequent attempts, if information matches, will be silently accepted
            - NOTE: if the key information changes, an error will be displayed and the attempt to connect will fail

212.3 Secure Shell (Advanced SSH - Using SSH Keys for Authentication)

- `ssh-keygen`
    - utility to generate a public/private key that can be used to authenticate a user on a remote system without providing an authentication password (if a password was not used during generation - something that is often done for two factor authentication)
    - `~/.ssh`
        - directory where public (id_rsa.pub) and private keys (id_rsa) are stored
            - NOTE: names of public/private keys can be changed during generation or renamed after (option during creation or on command line)
    - `-t [encryption]` • type of encryption to use (RSA is default)
- `ssh-copy-id`
    - method of copying a public key for use in remote authentication to a remote server
        - will automatically prompt for login and copy the key to the remote ~/.ssh/authorized_keys file
- `~/.ssh/authorized_keys`
    - file containing the hostname/IP and public key of systems that are authorization to provide a

key for access on login

- permissions on this file should be 640

- manual method
    - in the event the `ssh-copy-id` utility is not available, the following command can be used to copy a public key to a remote host
        - **Example** • `cat ~/.ssh/id_rsa.pub | ssh 10.1.0.100 'cat >> .ssh/authorized_keys` (you will be prompted on copy for the remote password)
- `ssh-agent`
    - allows you to associate a key with a session so that the first connection (if a passphrase is configured) will be the only prompted password
    - **Example** • `ssh-agent /bin/bash` (creates the shell with the agent associated)
      `ssh-add [~/.ssh/id_rsa]` (path not needed if default location and name)
      prompt for password - subsequent connections will not need it

212.4 Security Tasks

- system security
    - update your system - checking and installing daily updates is optimal, however, weekly is at a minimum, especially for key remote access utilities
- monitoring
    - periodic manual auditing provides valuable information, but automated monitoring and alerts on key system performance and access is essential to providing a secure system to your users
- objectives (covered elsewhere, mentioned here)
    - `nmap` • covered in LPIC-2 Exam 1
    - `nc` • covered in LPIC-2 Exam 1
    - `telnet` • covered in various places, valuable for testing service/port availability
    - `iptables` • covered in this course in '212.1 Configuring a Router'
- `fail2ban` (package to install on all distributions)
    - daemon that can scan log files, identifying IPs that are repeatedly attempted connections
    - `/etc/fail2ban/jail.conf`
        - primary configuration file for the fail2ban daemon
        - NOTE: recommended by the utility author NOT to use the system wide configuration file as it will be overwritten any time the tool is updated, the use of `jail.local` or `jail.d/`

`customization.local` is recommended for persistent settings

- banning of an IP happens on a service by service configuration basis (ssh for example)
- key settings
    - `bantime [#]` • how long (seconds) the host gets banned for
    - `maxretry [#]` • the maximum number of connection attempts before receiving a ban
    - `findtime [#]` • the amount of time the maxretry setting will use to track the maximum number (i.e. a value of 300 would mean 'N' failures within 5 minutes would result in a ban)
    - `enabled [true/false]` • enables the service jail for the section
    - `ignoreip [IP]` • white list (ignore) attempts from the indicated IP(s)
    - `action = [action command]` • run the indicated command when the ban is triggered (like sending root an email)
    - **Example** •

```
[sshd]
enabled = true
bantime = 300
findtime = 180
maxretry = 5
action = sendmail-whois[name=SSH,dest=root,sender=fail2ban@localhost]
```

        - this would configure the jail for `sshd` and enable it, setting the time an address is banned to 5 minutes, when 5 retries happen from that same address inside of any 3 minute window and notify root with an email containing the host information

- OpenVAS
    - specific configuration, setup and use is outside the scope of the LPIC-2 exam
    - Open Vulnerability Assessment System
    - several tools that provide information on vulnerabilities and alert on them
    - uses a database of known vulnerabilities for various accessibility applications
    - identifies and reports on security holes known or discovered
    - tests configuration settings for security issues
- Snort
    - specific configuration, setup and use is outside the scope of the LPIC-2 exam
    - snort is considered an IDS (Intrusion Detection System)

- helps to determine if someone/something is 'probing' or attacking your system
- network traffic analysis to determine if attacks are occurring
    - can be configured in three different modes
        1. packet logging
        2. real time network data view (sniffing)
        3. IDS

- services for alerting on security issues
    - CERT
        - Computer Emergency Response Team (specifically CERT Coordination Center or CERT-CC)
        - located at Carnegie Mellon University
        - provides tools for vulnerability assessment and analysis, comprehensive list of known vulnerabilities and attack vectors
        - often works with goverment organizations (or private institutions) regarding computer security
            - http://www.cert.org
    - US-CERT
        - the US government CERT
        - http://www.us-cert.gov
    - BugTraq
        - subscription e-mail program
        - created and paid for by the Security Focus organization
            - http://www.securityfocus.com
        - moderated (and detailed) mailing list for the discussion of (and announcements of new) security vulnerabilities
        - provides details on when discovered, what is affected (versions and types), what the vulnerability is and any known exploits

Appendix A - Sample Configuration Files

/etc/dhcpd.conf (Sample Configuration File from Package)

```
ddns-update-style interim;
ignore client-updates;
subnet 192.168.0.0 netmask 255.255.255.0 {
# --- default gateway
    option routers          192.168.0.1;
    option subnet-mask      255.255.255.0;
    option nis-domain       “domain.org”;
    option domain-name      “domain.org”;
    option domain-name-servers  192.168.1.1;
    option time-offset      -18000; # Eastern Standard Time
#   option ntp-servers      192.168.1.1;
#   option netbios-name-servers 192.168.1.1;
# --- Selects point-to-point node (default is hybrid). Don’t change this
unless
# -- you understand Netbios very well
#   option netbios-node-type 2;
    range dynamic-bootp 192.168.0.128 192.168.0.254;
    default-lease-time 21600;
    max-lease-time 43200;
    # we want the nameserver to appear at a fixed address
    host ns {
        next-server marvin.redhat.com;
        hardware ethernet 12:34:56:78:AB:CD;
        fixed-address 207.175.42.254;
    }
}
```

/etc/nsswitch.conf

```
# /etc/nsswitch.conf
#
# An example Name Service Switch config file. This file should be
# sorted with the most-used services at the beginning.
#
# The entry ‘[NOTFOUND=return]’ means that the search for an
# entry should stop if the search in the previous entry turned
# up nothing. Note that if the search failed due to some other reason
# (like no NIS server responding) then the search continues with the
# next entry.
#
# Valid entries include:
#
#   nisplus         Use NIS+ (NIS version 3)
#   nis         Use NIS (NIS version 2), also called YP
#   dns         Use DNS (Domain Name Service)
#   files           Use the local files
#   db          Use the local database (.db) files
#   compat          Use NIS on compat mode
#   hesiod          Use Hesiod for user lookups
#   [NOTFOUND=return]   Stop searching if not found so far
#
# To use db, put the "db" in front of "files" for entries you want to be
# looked up first in the databases
#
# Example:
#passwd:    db files nisplus nis
#shadow:    db files nisplus nis
#group:     db files nisplus nis
passwd:     files
shadow:     files
group:      files
#hosts:     db files nisplus nis dns
hosts:      files dns
# Example - obey only what nisplus tells us...
#services:   nisplus [NOTFOUND=return] files
#networks:   nisplus [NOTFOUND=return] files
#protocols:  nisplus [NOTFOUND=return] files
#rpc:        nisplus [NOTFOUND=return] files
#ethers:     nisplus [NOTFOUND=return] files
#netmasks:   nisplus [NOTFOUND=return] files
bootparams: nisplus [NOTFOUND=return] files
ethers:     files
netmasks:   files
networks:   files
protocols:  files
rpc:        files
services:   files
netgroup:   nisplus
publickey:  nisplus
automount:  files nisplus
aliases:    files nisplus
```

/etc/security/limits.conf

```
# /etc/security/limits.conf
#
#Each line describes a limit for a user in the form:
#
#<domain>        <type>  <item>  <value>
#
#Where:
#<domain> can be:
#        - a user name
#        - a group name, with @group syntax
#        - the wildcard *, for default entry
#        - the wildcard %, can be also used with %group syntax,
#                 for maxlogin limit
#
#<type> can have the two values:
#        - "soft" for enforcing the soft limits
#        - "hard" for enforcing hard limits
```

```
#
#<item> can be one of the following:
#        - core - limits the core file size (KB)
#        - data - max data size (KB)
#        - fsize - maximum filesize (KB)
#        - memlock - max locked-in-memory address space (KB)
#        - nofile - max number of open file descriptors
#        - rss - max resident set size (KB)
#        - stack - max stack size (KB)
#        - cpu - max CPU time (MIN)
#        - nproc - max number of processes
#        - as - address space limit (KB)
#        - maxlogins - max number of logins for this user
#        - maxsyslogins - max number of logins on the system
#        - priority - the priority to run user process with
#        - locks - max number of file locks the user can hold
#        - sigpending - max number of pending signals
#        - msgqueue - max memory used by POSIX message queues (bytes)
#        - nice - max nice priority allowed to raise to values: [-20,
19]
#        - rtprio - max realtime priority
#
#<domain>      <type>  <item>         <value>
#
#*               soft    core            0
#*               hard    rss             10000
#@student        hard    nproc           20
#@faculty        soft    nproc           20
#@faculty        hard    nproc           50
#ftp             hard    nproc           0
#@student        -       maxlogins       4
# End of file
```

/etc/pam.d/passwd

```
#%PAM-1.0
auth       include  system-auth
account    include  system-auth
password   substack system-auth
-password   optional    pam_gnome_keyring.so
```

/etc/pam.d/system-auth

```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_env.so
auth        sufficient    pam_unix.so try_first_pass nullok
auth        required      pam_deny.so
account     required      pam_unix.so
password    requisite     pam_cracklib.so try_first_pass retry=3 type=
password    sufficient    pam_unix.so try_first_pass use_authtok nullok
```

```
sha512 shadow
password    required      pam_deny.so
session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
session     [success=1 default=ignore] pam_succeed_if.so service in
crond quiet use_uid
session     required      pam_unix.so
```

/etc/samba/smb.conf (example configuration file)

```
# See smb.conf.example for a more detailed config file or
# read the smb.conf manpage.
# Run ‘testparm’ to verify the config is correct after
# you modified it.
[global]
    workgroup = SAMBA
    security = user
    passdb backend = tdbsam
    username map = /etc/samba/usermap
    printing = cups
    printcap name = cups
    load printers = no
    cups options = raw
[homes]
    comment = Home Directories
    valid users = %S, %D%w%S
    browseable = No
    read only = No
    inherit acls = Yes
[printers]
    comment = All Printers
    path = /var/tmp
    printable = Yes
    create mask = 0600
    browseable = No
[print$]
    comment = Printer Drivers
    path = /var/lib/samba/drivers
    write list = root
    create mask = 0664
    directory mask = 0775
[myshare]
    comment = This is our test share
    path = /myshare
    guest ok = no
    writeable = yes
```

/etc/named.conf (default - caching DNS server)

```
// named.conf
//
// Provided by Red Hat bind package to configure the ISC BIND named(8)
```

```
DNS
// server as a caching only nameserver (as a localhost DNS resolver
only).
//
// See /usr/share/doc/bind*/sample/ for example named configuration
files.
//
// See the BIND Administrator's Reference Manual (ARM) for details about
the
// configuration located in /usr/share/doc/bind-{version}/Bv9ARM.html
options {
    listen-on port 53 { 127.0.0.1; };
    listen-on-v6 port 53 { ::1; };
    directory   "/var/named";
    dump-file   "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    allow-query     { localhost; };
    /*
    - If you are building an AUTHORITATIVE DNS server, do NOT enable
recursion.
    - If you are building a RECURSIVE (caching) DNS server, you need to
enable
    recursion.
    - If your recursive DNS server has a public IP address, you MUST
enable access
    control to limit queries to your legitimate users. Failing to do so
will
    cause your server to become part of large scale DNS amplification
    attacks. Implementing BCP38 within your network would greatly
    reduce such attack surface
    +/
    recursion yes;
    dnssec-enable yes;
    dnssec-validation yes;
    /* Path to ISC DLV key */
    bindkeys-file "/etc/named.iscdlv.key";
    managed-keys-directory "/var/named/dynamic";
    pid-file "/run/named/named.pid";
    session-keyfile "/run/named/session.key";
};
logging {
        channel default_debug {
                file "data/named.run";
                severity dynamic;
        };
};
zone "." IN {
    type hint;
    file "named.ca";
};
include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
```

/etc/named.conf (configuration for full ‘mydomain.com’ example DNS server)

```
//
// named.conf
//
// Provided by Red Hat bind package to configure the ISC BIND named(8)
DNS
// server as a caching only nameserver (as a localhost DNS resolver
only).
//
// See /usr/share/doc/bind*/sample/ for example named configuration
files.
//
options {
    listen-on port 53 { 127.0.0.1; any; };
    listen-on-v6 port 53 { ::1; any; };
    directory   “/var/named”;
    dump-file   “/var/named/data/cache_dump.db”;
    statistics-file “/var/named/data/named_stats.txt”;
    memstatistics-file “/var/named/data/named_mem_stats.txt”;
    allow-query     { localhost; 10.1.0.0/24; };
    /*
    - If you are building an AUTHORITATIVE DNS server, do NOT enable
recursion.
    - If you are building a RECURSIVE (caching) DNS server, you need to
enable
    recursion.
    - If your recursive DNS server has a public IP address, you MUST
enable access
    control to limit queries to your legitimate users. Failing to do so
will
    cause your server to become part of large scale DNS amplification
    attacks. Implementing BCP38 within your network would greatly
    reduce such attack surface
    */
    recursion yes;
    dnssec-enable yes;
    dnssec-validation yes;
    /* Path to ISC DLV key */
    bindkeys-file “/etc/named.iscdlv.key”;
    managed-keys-directory “/var/named/dynamic”;
    pid-file “/run/named/named.pid”;
    session-keyfile “/run/named/session.key”;
};
logging {
        channel default_debug {
                file “data/named.run”;
                severity dynamic;
        };
};
zone “.” IN {
    type hint;
    file “named.ca”;
};
zone “mydomain.com” IN {
    type master;
    file “fwd.mydomain.com.db”;
    allow-update { none; };
};
zone “0.1.10.in-addr.arpa” IN {
    type master;
    file “0.1.10.db”;
    allow-update { none; };
};
include “/etc/named.rfc1912.zones”;
include “/etc/named.root.key”;
```

/var/named/named.localhost

```
$TTL 1D
@  IN  SOA  @ rname.invalid. (
              0       ; serial
              1D      ; refresh
              1H      ; retry
              1W      ; expire
              3H )    ; minimum
NS      @
A       127.0.0.1
AAAA    ::1
```

/var/named/fwd.mydomain.com.db (forward lookup zone file)

```
$TTL 86400
@   IN  SOA     named.mydomain.com. root.mydomain.com. (
            10020       ;Serial
            3600        ;Refresh
            1800        ;Retry
            604800      ;Expire
            86400       ;Minimum TTL
)
;Name Server Information
@       IN  NS      named.mydomain.com.
;IP address of Name Server
primary IN  A       10.1.0.100
;A - Record HostName To Ip Address
named   IN  A       10.1.0.100
;CNAME record
dns IN CNAME    named.mydomain.com.
```

/var/named/0.1.10.db (reverse lookup zone file)

```
$TTL 86400
@   IN  SOA     named.mydomain.com. root.mydomain.com. (
            10020       ;Serial
            3600        ;Refresh
            1800        ;Retry
            604800      ;Expire
            86400       ;Minimum TTL
)
;Name Server Information
@ IN  NS      named.mydomain.com.
;Reverse lookup for Name Server
101 IN  PTR named.mydomain.com.
;PTR Record IP address to HostName
102 IN  PTR server2.mydomain.com.
```

LDIF Sample - Top Level Directory

```
dn: dc=mydomain,dc=com
dc: mydomain
description: creating my dc
objectClass: dcObject
objectClass: organization
o: mydomain,llc.
```

LDIF Sample - Creating an Organizational Unit

```
dn: ou=people, dc=mydomain,dc=com
ou: people
description: Everyone in the company
objectclass: organizationalunit
```

LDIF Sample - Adding a Single Record

```
dn: cn=Robert Smith,ou=people,dc=my-domain,dc=com
objectclass: inetOrgPerson
cn: Robert Smith
cn: Robert J Smith
cn: bob  smith
sn: smith
uid: rjsmith
userpassword: myPwd123
carlicense: ABC 123
homephone: 555-111-3322
mail: r.smith@mydomain.com
mail: rsmith@mydomain.com
mail: bob.smith@mydomain.com
description: Nice Guy
ou: Human Resources
```

LDIF Sample - Adding Multiple Records

```
dn: cn=Clark Kent,ou=people,dc=my-domain,dc=com
objectclass: inetOrgPerson
cn: Clark Kent
cn: Clark G Kent
sn: Kent
uid: ckent
userpassword: s00perm@n
carlicense: HISCAR 124
homephone: 555-111-2223
mail: c.kent@mydomain.com
mail: ckent@mydomain.com
mail: clark.kent@mydomain.com
ou: Sales
## add another entry to our OU
dn: cn=Bruce Wayne,ou=people,dc=my-domain,dc=com
objectclass: inetOrgPerson
cn: Bruce Wayne
sn: wayne
uid: bwayne
userpassword: IAmNotBatman
carlicense: BATMAN 123
homephone: 555-111-2225
mail: b.wayne@example.com
mail: bwayne@example.com
mail: bruce.wayne@example.com
ou: IT
```