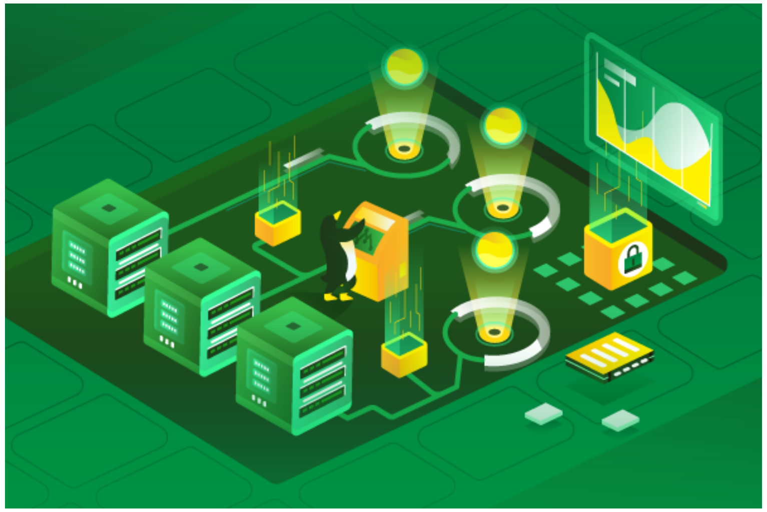

LPIC3-303 Study Guide

Stosh Oldham
stosh@linuxacademy.com
February 25, 2019

Contents

About LPIC3-303: Security

- Signing Up for the Exam
  - Purchase exam voucher with PearsonVUE at **http://www.pearsonvue.com/lpi/**
  - Note Linux Academy frequently provides a discount code for the exam!

    See **https://linuxacademy.com/cp/courses/lesson/course/2933/lesson/2/module/252** for the latest information!
  - LPIC-3 303 Security is Version 2.0 as of this writing.
  - Exam code is 303-200.
- About the Exam
  - Test is 60 questions.
  - Questions are composed of multiple choice as well as fill-in-the-blank.
  - You are given 90 minutes to take the test.
  - Questions come from a large question pool, which means you may see different questions for each attempt.

Topic 325 Cryptography

325.1 X.509 Certificates and Public Key Infrastructures

Cryptography Concepts

- Uses of cryptography
  - Authentication
  - Integrity
  - Data Encryption
- There are two primary elements of cryptography
  - Key:

    Used to encrypt data

    Must be secret
  - Algorithm:

    Method of encryption

    May be public

    Examples: 3DES(old), blowfish, AES
- Encryption
  - A cipher (or algorithm) is used to scramble information
  - The ciphertext may be deciphered (or unencrypted) with a key
  - There are two types of encryption in modern cryptography

- Asymmetric Encryption:
    - Also known as public key cryptography
    - Uses 2 keys:
        - One for encryption (public key)
        - One for decryption (private key)
    - Common Algorithms: RSA, DSA, PKCS
    - Used for digital signatures, key distribution, digital certificates
- Symmetric Encryption:
    - The same key may be used to both encrypt and decrypt information
    - Both parties must know the key
    - Algorithms: Blowfish, AES
    - Generally faster than asymmetric encryption
    - Used for data integrity

- Hashes
    - Converts a string of any length to an output string of a fixed length:
        - Each string provides a unique hash
        - Hashing is generally one way (no means of getting the original value)
        - Salt may be used to improve security:
            - Salt is an additional value (typically random) added to the information making it harder to crack
        - Common algorithms: `crc-32` (insecure), `md5`, `sha-1` (most common)
- Hashing utilities
    - `md5sum`: Creates a hash based on input
    - `openssl dgst`

PKI and Trust chains

- **PKI** stands for **Public Key Infrastructure**
    - Defined by X.509 standard and RFC5280
- It is important to be familiar with the fields of the x509v3 standard
- Specifically `CA: True` and `pathlen`
    - `CA: True` indicates the certificate is a CA
    - `pathlen 0` indicates the number of levels down that a certificate signed by the CA cert in question is permitted

        A value of `0` indicates that certificates signed by this CA may not sign any certificates

    - Seeing subject alternative names (SANs) in x509v3:

```
 Requested Extensions:
X509v3 Subject Alternative Name:
DNS:example.com, DNS:www.example.com
```

        SANs are essentially aliases for a domain name covered by a single certificate

        It is important to note the format returned in the x509 data (noted above)

    - Uses public key encryption

        There is a public and private key

        Uses of PKI:

            Proof of identity

            Encryption

            Proof of data integrity

    - PKI requires uses of a Certificate Authority or CA

- The CA is a trusted third party that validates the authenticity of a public key
  - CAs also maintain and publish CRLs (Certificate Revocation Lists)
  - CRLs are gradually being replaced by OCSP or Online Certificate Status Protocol
  - OCSP is only supported in TLS1.2 and newer
- More information on OCSP is provided in section 325.2 of this guide
  - To create a new certificate, a Certificate Signing Request (CSR) is generated:
    - The CA signs the CSR to issue the new certificate
    - You may sign your own CSR using a tool such as `openssl` which would generate a "self-signed certificate":
      - In this case, you are acting as your own CA

Request, Sign, and Manage Certificates

- Using `openssl`
- Use `openssl genrsa` to make a private key
  - You may specify a key size, an encryption algorithm (this is recommended), and an output file (or use command line redirection)
  - Using encryption on a key requires a passphrase which will be gathered by a prompt
  - `openssl genrsa -des3 -out -mykey.key 2048`
  - For additional configuration, you may use additional parameters (which are optional): `config openssl.cnf`
  - How to issue a Certificate Signing Request (CSR):
    - Generate a key with `openssl genrsa` (see above)
    - Use `openssl req -new -key mykey.pem -out mycsr.key` to make CSR
    - The `-new` flag indicates the creation of a new CSR

        - If `-key` is not provided, a key will be automatically generated
        - The `-out` flag takes the filename of the newly created CSR
    - Using `openssl req` on a CSR with `-noout` and `-text` will allow a viewing signing request
- Certificate file formats
    - `openssl` creates keys and CSRs in PEM format by default
    - PEM stands for Privacy-enhanced Electronic Mail
    - PEM is a simple base64 encoded ASCII file
    - There are a few other formats that may be required depending on your need:
        - DER stands for Distinguished Encoding Rules
        - DER is a binary form of ASCII PEM
        - PKCS stands for Public Key Cryptography Standards
        - There are many standards named by number (i.e., pkcs#7) but only a couple have file formats:
            - P7B/PKCS#7: base64 encoded ASCII popular in windows
            - PFX/PKCS#12: A binary format capable of storing keys, certs, and intermediary certs together
- Various `openssl` sub-commands can do conversion as noted below:
    - DER to PEM:

```
openssl x509 -inform der -in certificate.cer -out certificate.pem
```

    - PEM to DER:

```
openssl x509 -outform der -in certificate.pem -out certificate.der
```

    - p7b/pkcs#7 to PEM:

```
openssl pkcs7 -print_certs -in certificate.p7b -out certificate.pem.
```

    - PEM to p7b/pkcs#7:

```
openssl pkcs7 -print_certs -in certificate.p7b -out certificate.cer
```

- pfx/pkcs#12 to PEM:

  Get private key: `openssl pkcs12 -in filename.pfx -nocerts -out key.pem`

  Get certs: openssl `pkcs12 -in filename.pfx -clcerts -nokeys -out cert.pem`

- Recall that, by default, `openssl` works in PEM format
  - See x509, rsa, pkcs7, and pkcs12 man pages for more information

Operating a Certificate Authority

- `openssl` for a Certificate Authority
  - CAs must have their own keypair that may be generated using a tool such as `openssl`:

    `openssl genrsa -des3 -out -root-ca.key 2048`

    `passphrase` is required for CA

    Use the following to make a self-signed cert:

    ```
     openssl req -utf8 -new -key <private_key_file> -x509 -days 365 \
    -out <output_file> -set_serial 0
    ```

    `-new` is used for new CSRs

    `-utf8` is used for `utf8` string formatting - ASCII is the default

    `-key` <\private_key_file> is the private key to use for the new request

    `-x509` is what makes the command issue a self-signed certificate. Without it, a CSR is generated

    `-days` is the number of days the certificate should be valid and may only be used in conjunction with the `-x509` option in this use case

    `-set_serial` allows a serial number to be set for a self-signed certificate

    Must provide serial for CA certificate

  - When run, the command prompts for the password of the private key as well as a number of `x509` standard piece of information
  - Common name is the only mandatory field
- Basic `openssl ca` set up
  - There are a couple required files that `openssl ca` must have to work (creation commands noted below):
    - `echo 00 > /etc/pki/CA/serial`
    - `touch /etc/pki/CA/index.html`
  - Signing a Certificate Signing Request (CSR):
    - Use `openssl ca` command
    - `openssl -csr input_file -crt output_file`
  - Revoking a signed key:
    - Option 1: Publish a Certificate Revocation List (CRL)
      - CRLs must be regularly checked by clients, and this is outside of CA control
    - Option 2: Online Certificate Status Protocol (OCSP)
      - Using OCSP, a CA maintains a database and responds to OCSP requests from end users
      - An example OCSP server implementation is *Let's Encrypt's Boulder* server
      - Security can be improved by issuing shorter certificate expiration dates
- Alternative tool: `genkey` on RedHat
  - Simpler than `openssl`: uses TUI
  - See `genkey --help` for a list of subcommands

325.2 X.509 Certificates for Encryption, Signing, and Authentication

SSL, TLS, and Apache HTTPD Server

- Secure Sockets Layer (SSL)
    - The latest version is SSLv3, which has been deprecated
    - There are known vulnerabilities in SSL that make it insecure
    - Most browsers will issue a warning when connecting over SSL to a web host
- Transport Layer Security (TLS)
    - TLS is the successor to SSL
    - TLS 1.0 served as an effective upgrade to SSLv3
    - TLS 1.2 is the current standard
    - TLS 1.3 has been purposed as an Internet Standard
- SSL and TLS aim to address the issue of secure communication over a public network
    - Transport layer security must achieve the following:

        Securely encrypt exchanged data

        Authenticate at least one party

        Ensure data integrity

        Prevent replay attacks
    - TLS uses PKI to agree on a shared secret and then establish a shared key for symmetric encryption of exchanged data
    - PKI can authenticate the identity of a party by means of a trust third party, the Certificate Authority
    - As newer versions of the protocols are released:

        Less secure ciphers are de-supported

Vulnerabilities are patched (example: POODLE, BEAST)

- Man in the Middle Attacks
  - A Man in the Middle Attack is an attack whereby a third party manages to come between a user and an application and is able to intercept and sometimes alter data passing between the parties
  - The attacks typically target sensitive information such as an account number or user login credentials
  - How to mitigate:

    Keeping system software updated

    De-support vulnerable technologies such as TLS 1.0

    Disable weak ciphers

    Always encrypt all network traffic

    Some responsibility still rests with end users to fully avoid this style attack

mod_ssl with Apache

- Install `mod_ssl` package with Apache (comes by default with many apache `httpd` distributions)
- Configuration file is `ssl.conf` file
  - The Concept:

    Specify a keypair to be used by Apache to establish transport layer security

    A few simple directives are used to provide the necessary keys and certificates

    A default virtual host is configured to use secure communication with `mod_ssl`

    The `mod_ssl` module does the work of presenting the keypair to clients and encrypting traffic over TLS
  - Key directives for HTTPS configuration:

    Enhancing cipher strength:

      Apache supports a wide range of ciphers for various levels of compatibly

- Cipher support may be tuned to enhance or loosen security as necessary
- Enable only the strongest ciphers in `httpd` config:
  - `SSLCipherSuite HIGH:!aNULL:!MD5`
  - You may also be more granular in cipher support.

    ```
    SSLCipherSuite RC4-SHA:AES128-SHA:HIGH:!aNULL:!MD5
    SSLHonorCipherOrder on
    ```

  - SSLCipherSuite may be tunned as necessary but, naturally, you should always use the strongest ciphers possible
- Configuring server-side certificates:
  - SSLCertificateFile: Public key certificate given to users:
    - Note: As of `httpd 2.4.8`, the intermediate CA certificates may also be included in this file
    - The Certificates should be provided from *root* to *leaf* (highest level CA at the bottom)
    - This new functionality obsoletes SSLCertificateChainFile
  - SSLCertificateKeyFile: Private key for a webserver

Configure client certificate authentication

- The Concept:
  - Have web clients authenticate to your web server using a provided certificate
  - The certificate is generated and signed by the local web server
  - There are a few key directives that handle ensuring clients are authenticated prior to accessing sensitive content
- Key directives for client certificate authentication:

```
 SSLVerifyClient require
SSLVerifyDepth 1
SSLCACertificateFile "conf/ssl.crt/ca.crt"
```

SSLVerifyClient: Instructs server to verify client certificate

SSLVerifyDepth: Number of intermediaries (1 for local CA signed)

SSLCACertificateFile: CA certificate to check against

OCSP Stapling

- The concept:
    - By default, OCSP requires that the client checks to see if a certificate has been revoked with the CA who signed it
    - When OCSP stapling is enabled, the web server maintains the current OCSP response from the CA and sends the response instead of the client having to contact the CA:
    - This requires minimal overhead on the web server
    - OSCP has the following benefits
        - Reduced load on CAs
        - Faster page load times for clients
        - Improved quality control on page load times for web administrators
- In a global scope with general SSL configuration, use the following directives:

```
 SSLUseStapling On
SSLStaplingCache "shmcb:logs/ssl_stapling(32768)"
```

- Note that the path on the SSLStaplingCache directive should match the one on the SSLSessionCache directive
- The SSLStaplingCache path is relative to ServerRoot

Verify OCSP stapling is enabled: `openssl s_client -connect \` `www.example.com:443 -status -servername www.example.com`

- Using Server Name Indication (SNI)

  The Concept:

    Allows you to host multiple SSL certificates from the same socket connection (IP with Port)

    Simply specify named based virtual hosts using the `ServerName` directive and follow it with the Key SSL configuration directives (noted earlier)

  Requires Apache v2.2.12 and OpenSSL v0.9.8j and later

  Some browsers (especially older versions) do not support SNI which may require some contingency planning

  The directive `SSLStrictSNIVHostCheck` may be used to control if non-SNI clients are able to access a name-based virtual host

- HTTP Strict Transport Security - HSTS

  The Concept:

    HSTS is a means of a web server indicating that all content will only be served over HTTPS

    Without special configuration, web servers will send traffic over HTTP or HTTPS based on client request

    HSTS involves providing a special header information indicating that it only will serve content over HTTPS

    Any non-HTTPS requests generally receive a 301 redirect to the same resource using HTTPS

  Implementation:

    Requires a properly secured website using `mod_ssl`

    Requires `mod_headers.so` and `mod_rewrite.so` (typically installed by default)

    Configure Apache in the appropriate context to send appropriate header information:

```
# Use HTTP Strict Transport Security to force client to use secure \
 connections only Header always set Strict-Transport-Security \
 "max-age=300; includeSubDomains; preload"
```

The Max-age directive is required and specifies the number of seconds the site should be regarded as a known HSTS host

When implementing HSTS, start with a lower max-age and work up to a 2-year max-age over a few weeks

This will allow fixes to be issued if needed during the transition to HSTS

`includeSubDomains` is only needed if the domain has sub-domains

A wild-card SSL certificate is necessary to support sub-domains

The preload directive tells browsers to treat the host as a known HSTS host and to not make non-HTTPS requests

The preload directive should only be added after a successful testing period with HSTS

Configure Redirect to force HTTPS when HTTP request is made:

```
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI} [R=301,L]
</IfModule>
```

Troubleshooting with `openssl`

- Using `openssl` for troubleshooting
    - Use `openssl verify -verbose <certificate>` to validate chain of trust

- Use `openssl s_client`
  - Parameters:
    - `-connect <host>`: host to connect to
    - `-showcerts`: displays certificate
    - `-servername`: TLS SNI name sent with ClientHello
  - Use `openssl x509 -text -in <cert>` to see contents of CA cert (or any cert for that matter)

325.3 Encrypted File Systems

Creating Encrypted Volumes

- Concepts
  - Disk encryption aims to protect data stored on disk through encryption
  - This protects confidential data
  - There are two primary kinds of disk encryption:
    - Block Device Encryption:
      - Encrypts whole block devices
      - Block device encryption tools: `dm-crypt` or LUKS (Linux Unified Key Setup)
    - File System Encryption:
      - Encrypt data at the files system level as opposed to the device
      - File System Encryption tools: eCryptFS and EncFS
  - File System encryption with eCryptfs
    - `Ecryptfs` provides an easy-to-use means of file system encryption

- Using the `ecryptfs` package, you only need to run a `mount` command to create an encrypted directory
  - Run `mount -t ecryptfs /src_dir /dest_sir`
  - This command will prompt for encryption settings and then create the `<dest_dir>` as an encrypted directory
  - The settings may be supplied using the `-o parameter=value` format with the command as well
  - The command does require `root` privilege
  - Once mounted, the directory may be used in a typical fashion
  - Once the directory is unmounted, the data becomes unreadable
- There are a number of commands included in the `ecryptfs-utils` package that provides enhanced functionality
  - There is a suite of commands that are usable by non-super users for creating and using an encrypted `~/Private` directory
    - Setup with `ecryptfs-setup-private`
    - Mount and unmount using `ecryptfs-mount-private` and `ecryptfs-unmount-private` is used to add an eCryptfs mount passphrase to the kernel keyring
  - `ecryptfs-add-passphrase`
  - Use `ecryptfs-manager` to manage keys
  - `ecryptfs-stat` can provide information on files encrypted using ecryptfs
- man `mount.ecryptfs`, `umount.ecryptfs`
- PAM integration is supported with ecryptfs using the module `pam_ecryptfs.so`
- PAM in general is covered in greater detail in section 326.3 of this Study Guide
- This allows for automatic mounting of encrypted volumes on login
- To configure the integration:
  - File system options and mount points must be supplied in `/etc/fstab`

            - The ecryptfs mount passphrase must be stored in `~/.ecryptfs/wrapped-passphrase`
            - Edit `/etc/pam.d/login`:
                - `auth required pam_ecryptfs.so unwrap`
            - Edit `/etc/pam.d/common-session`:
                - `session optional pam_encryptfs.so unwrap`
        - See `man pam_ecryptfs` for additional information
    - The `encfs` package provides similar functionality to `ecryptfs` but it is designed to be used by non-superusers
        - Create and mount an encrypted repository, run `encfs ~/.name ~/name`
            - `.name` and `name` may be any current directories that your current user has access to
            - Work with `~/name` as a normal directory; `~/.name` is where the encrypted data is stored
            - Similar to `ecryptfs`, you are prompted for input during setup
            - Note that absolute file paths are required
        - Use `encfsctl passwd ~/name` to change the passphrase if desired
        - Use `fusermount -u ~/name` to unmount and secure data
        - See `man encfs` for more information

Working with LUKS

- LUKS comes with the `cryptsetup` package
- LUKS is essentially a wrapper around `dm-crypt`
- It is possible to use plain `dm-crypt` for block device encryption:
    - LUKS provides some safeguards around `dm-crypt`
    - See `man cryptsetup` for more information

- Encrypting a block device with LUKS:
  - First, the block device (`ex /dev/sdb1`) must be formatted using the LUKS format

    ```
    cryptsetup luksFormat /dev/sdb1
    ```

  - Once the volume is created, it must be mapped with the device mapper

    ```
    cryptsetup luksOpen /dev/sdb1 secret
    ```

    The parameter "secret" is the name used by device mapper to map our device
  - You can use the `status` and `luksDump` subcommands of `cryptsetup` to view information regarding the device
  - The encrypted volume requires a file system for general use.
  - `mkfs.ext4 /dev/mapper/secret`
  - Once the file system is in place, the volume may be mounted
  - `mount /dev/mapper/secret /mnt/classified`
  - Two options of persistent mount:

    Option 1: Use LUKS key
    1. Create LUKS key file:

       ```
       dd if=/dev/urandom of=/root/lukskey bs=4066 count=1
       chmod 600 /root/lukskey
       cryptsetup luksAddKey <luks_volume> /root/lukskey
       ```

       Prompted for the passphrase to protect the key
    2. Edit `/etc/crypttab`:

       ```
       name_of_encrypted_dev name_of_underlying_dev name_of_keyfile
       ```

    1. Edit `/etc/fstab` to map mapper device as normal file system.

       Note: `mount -a` does not process `/etc/crypttab`
    2. Additional `cryptsetup` commands to be familiar with:
       1. `cryptsetup luksAddKey`: Add key to luks device
       2. `cryptsetup luksRemoveKey`: Remove a supplied key from luks device

3. `cryptsetup luksKillSlot <device> <key slot number>`: Remove a key by it's position in luks meta data
4. `cryptsetup luksChangeKey`: Change existing passphrase

See the following for more detail:

`man crypttab`

`man cryptsetup`

Option 2: Enter pass phrase at prompt on boot

Follow the same steps as option 1 but do not create the key in `/etc/crypttab`

- The `cryptmount` package is also able to work with LUKS
  - It is noted in the man page for cryptmount that cryptsetup is still recommended when only working with block devices
  - Cryptmount is an alternative to cryptsetup
  - The `cryptmount-setup` command may be used by non-superusers to create encrypted file systems on plain files
  - The key benefit of cryptmount is that it allows users to mount and unmount their own encrypted file systems with a few caveats

    They must have a key or passphrase to access the file system

    There is some setup required by the `root` user
  - See the man pages for cryptmount and cmtab for more information

325.4 DNS and Cryptography

Working with DNS

- Domain Name System
  - The Concept:
    - DNS resolves a hostname to an IP address.
    - DNS worldwide hierarchy beginning with `root` servers (com, edu, org)
    - Local name servers will use a forwarder to look up hostnames if possible
    - If no forwarder is configured, a name server will reach out to a root name server which will reach out to the intermediate name servers to resolve a particular hostname
    - Zones and Resource Records (RRs)
      - A zone is a particular set of DNS data at some subdomain level (i.e., example.com)
      - RRs are the kind of resources that can be mapped with DNS
      - RRs consist of a type and a series of labels that make up domains, subdomains, host, etc.
  - EDNS: Extension mechanisms for DNS (RFC 2671)
    - Addresses backward compatibility with older versions of DNS protocol
- Working with BIND
  - BIND is a DNS server
  - BIND is configured using the file `named.conf` which is typically in `/etc`, but may vary by build
  - Securing BIND
    - A Transaction Signature (TSIG) is used for secure zone transfer in BIND
      - With TSIG, a shared secret key is used to verify record information

The named process may be run in a chroot jail, which isolates the process from all files on the system except those that are essential to the named's operation

Configuration directives in `named.conf`:

  Allow query:

    Semicolon delimited list of hosts or networks which query this DNS server

    You may also disallow certain networks or hosts with the `!` symbol

  recursion - A DNS server can reach out to other DNS servers for information

  allow transfer - Controls which DNS servers may collect zone information from this server

RNDC - Remote name server daemon control

  Uses `/etc/rndc.conf` as config file

  RNDC shares a secret key with BIND to prevent unauthorized access

  A default key is generated on install in `/etc/rndc.key`

  A new key may be generated using `rndc-confgen -a`

DNSSEC (an extension of TSIG) which will be covered in more detail late in this guide

- Tools for working with DNS:
  - dig - A DNS lookup utility:

    `dig @server name type`

    Server is the name or IP of the name server to query:

      Dig uses `/etc/resolv.conf` if no server is specified

    Name is the RR to be looked up such as a hostname

    Type indicates the type of query such as an A record, MX record, CNAME, etc.

      Records are looked up by default
  - delv - An newer (BIND 9.10 and later) enhanced version of dig that supports DNSSEC

    - `openssl`

Securing DNS with DNSSEC

- What is DNSSEC?
  - The concept

    DNSSEC or Domain Name System Security Extension guarantees the authenticity of Zone Transfers and RR lookups

    This is done by using public and private keys to sign data digitally

    Having digitally signed data prevents attackers from spoofing the IP or hostname of your name server to provide falsified information
  - Working with DNSSEC

    DNSSEC is enabled in `named.conf` with the following directives:

    `dnssec-enable yes;`

    `dnssec-validation yes;`

    RRs related to DNS:

    Resource Record Signature (RRSIG) which is used to authenticate records

    DNSKEY - A DNSSEC Public Key

    DS - Contains the hash of a DNSKEY record including the Key Signing Key

    NSEC or NSEC3 - RR for authenticated and explicit denial of existence

    Zone Singing Keys (ZSKs) must be created for each zone

    The Zone signing key is used by resolvers to authenticate the RRs in a particular zone

    The ZSK should be kept in `/var/named` or wherever your zone files are kept

    The ZSK record will need to be added to the zone file it authenticates

A key pair is created with the command: `dnssec-keygen -a RSASHA1 -b 1024 -n ZONE myzone.example.com`

- `-a` is the algorithm or cipher
- `-b` is the key size in bits
- `-n` is the nametype which may be `ZONE`, `HOST`, or `ENTITY`
- The final parameter is the zone file

A KSK, or Key Signing Key, is used to create a digital signature for ZSKs

- The KSK is what authenticates ZSKs for your domain
- The KSK must also be created using `dnssec-keygen`
- `dnssec-keygen -a RSASHA1 -b 4096 -n ZONE -f KSK myzone.example.com`
  - `-f` is the flag field which is primarily for setting KSK value in the flag field
- The KSK record must also be added to each zone file

Use `dnssec-signzone` command to sign the zone with the generated keys creating an RR signature or RRSIG

- It may also generate signed NSEC records
- `dnssec-signzone -e +3024000 -N INCREMENT myzone.example.com`
- `-e`: Time when the RRSIG will expire
- `-N`: Used to control the SOA serial number; in this case, it is incremented
- The command should be set to run monthly to resign the zone
- The `dnssec-settime` command can be used to manage the validity period of a given key
- Key rollover is necessary for optimal security
- The `dnssec-signzone` command creates a `myzone.example.zone.signed` zone file that should be used instead of the original, unsigned, file

            You must contact your registrar so that they will create a DS (or delegation signer) record in order for you to use DNSSEC

            The command `dnssec-dsfromkey` generates a DS RR for a provided key

                Resolvers hash your public KSK and compared it against the published DS record from your registrar

                This authenticates your KSK which will, in turn, be used to authenticate you ZSKs

                ZSKs are then used to authenticate your Resource Record Sets

                There are DS records for each level of the hierarchy up to the root name server which has an implicitly trusted KSK by virtue of a key signing ceremony

                This creates a chain of trust

        DO bit

            Stands for DNSSEC OK

            If set to 1, it indicates that the resolver supports DNSSEC security RRs

        AD bit

            Stands for Authenticated Data

            Set if all data returned by a given DNS server is authentic

        See RFC4035 and RFC4033 for more detail

- What is DANE?
    - DANE stands for DNS-Based Authentication of Named Entities
    - The Concept:

        DANE aims to bind public keys to DNS names in order to increase the rigor by which they are validated

        DANE accomplishes this by use of a TLSA RR in DNS

        TLSA RRs are used to associate a TLS server certificate to a domain name where the record is found

TLSA RRs add further rigor to the PKI process by applying a check against a presented public key in the form of a DNS validation

See RFC6698 for more detail

- Example TLSA RRs

```
  _443._tcp.www.example.com. IN TLSA (
  1 1 2 92003ba34942dc74152e2f2c408d29ec
        a5a520e7f2e06bb944f4dca346baf63c
        1b177615d466f6c4b71c216a50292bd5
        8c9ebdd2f74e38fe51ffd48c43326cbc )
```

- Note the port and protocol designation in the label.
- `_PORT._PROTOCOL.``
- The TLSA is not limited to https: `_25._SMTP.mail.example.com...`

Topic 326 Host Security

326.1 Host Hardening

Kernel Security

- Disabling unnecessary software
  - There are two options for managing services on a Linux system:

    The new way - `systemctl`

    `systemctl <enable | disable | mask> <name>`:

    Enable starts service on boot

    Prevents the system from starting on boot

    Mask maps target to `/dev/null` so that unit activation is impossible
    `systemctl enable` will not work on a masked unit

    `systemctl list-units <pattern>`:

    Show all units matching pattern

    Example showing all services:

    `systemctl list-units *.service`

    The older way - `chkconfig`

    `chkconfig <name> <on|off>`:

    `on` enables service for boot

    `off` disables service for boot

`chkconfig --list`

Outputs a list of services and the runlevels for which they are enabled

The `chkconfig` command has been replaced with `systemctl` on most modern distributions of Linux

- Using the commands for listing active services, you can determine which services are configured to run and disable those that should not be running

  Commonly disabled services for servers:

  `avahi-daemon`

  `cups`

  `atd`

- Limiting resource usage
  - The Concept:

    The `pam_limits.so` module is capable of limiting user access to system resources such as file handlers, active processes, open files, and other such resources

    There are both soft and hard limits:

    A regular user may temporarily bypass soft limits

    Hard limits may only be raised by `root` and are enforced by the kernel

    A user or group may apply limits

    You will find the `pam_limits.so` module enabled by default on most systems in `/etc/pam.d/system-auth`

    `session required pam_limits.so`

    See `man pam_limits` for more detail

    More details on PAM may be found in section 326.3

  - The `ulimit` command can be used to adjust resource limitations on the fly

    Limits that may be controlled via the `ulimit command`.

Here are some of the more common options for `ulimit`:

- `-a`: All current limits are reported
- `-f`: The maximum size of files written by the shell and its children
- `-t`: The maximum amount of cpu time in seconds
- `-u`: The maximum number of processes available to a single user
- `-T`: The maximum number of threads

- Persistent limits may be set using `/etc/security/limits.conf`

  The file is self documented

  The general format is as follows:

```
\#<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
```

  See `man limits.conf` for more detail.

- Drop unnecessary capabilities
  - Tuning the kernel with `sysctl`

    The `sysctl` command allows the setting of various kernel parameters which allow kernel configuration

    `/proc/sys` contains a number of directories and files which are representative of currently set kernel parameters

    The directory structure provides a clue as to what the kernel parameter name is

Example: `/proc/sys/kernel/hostname` is `kernel.hostname`

Using the `sysctl` command, we can interact with the kernel parameters

`sysctl -a`

Lists all kernel parameters

`-a`: Reads values from memory

`-ar <token>`: May be used to prune results for a given token

`sysctl -w kernel.hostname=mynewhostname`

`-w`: Writes values in memory (does not persist reboot)

`sysctl -p`

`-p` reloads from a specified file or the default file (`/etc/sysctl.conf`) if none is given

Key value pairs may be stored in `/etc/sysctl.conf` to be loaded on boot

See `man proc` and `man sysctl` for more detail

- Managing ASLR

The Concept:

ALSR stands for Address Space Layout Randomization

Each time a program runs, it uses an arbitrary space in memory

This behavior is not the default without ALSR

Programs must be capable of running under ALSR (Most modern programs are)

Setting or Un-setting ASLR:

ALSR is controlled by the kernel parameter `kernel.randomize_va_space`

This value may be controlled using `sysctl` or in /`etc/sysctl.conf` (see notes earlier in this section)

A value of `2` means ALSR is fully functional

A value of `1` means ALSR is operating in conservative mode

A value of `0` means ALSR is disabled

- NX bit

  The Concept:

    Most all modern CPUs have a means of securing systems through executable space protection

    This manifests on the system as the NX bit being set

    Having the NX bit set prevents execution from a specific memory area that is exclusive to simple data storage

    By limiting executable memory space, malicious programs have a harder time executing arbitrary code

  Verify NX bit set

    As NX is at the CPU level, you must look at CPU information to verify it:

    ```
    grep -Fw nx /proc/cpuinfo
    ```

  Exec-Shield is a software implementation of the NX bit for CPUs that do not support such technology

- Kernel-based network security

  ICMP may be completely disabled by setting `net.ipv4.icmp_echo_ignore_all` to `1`

  It is also possible to be more granular with ICMP tuning

  You might only want to prevent response to ICMP echo broadcasts

    Setting the `net.ipv4.icmp_echo_ignore_broadcasts` to `1` prevents an ICMP echo response from the kernel

    This value may be controlled using `sysctl` or in /`etc/sysctl.conf` (see notes earlier in this section)

- Chroot Environments

  - The concept:

    The `chroot` command sets the root of the file system to a predesignated location in the normal file system for a provided process

    The process is confined exclusively to the new root until the process terminates

If that process has escalated privileges or access to block devices, it is possible the process can escape the chroot environment

- Important Notes on chroot environments:

  Take care when setting up a chroot environment, as a jailed process requires access to all necessary files while in the chroot environment

  While symbolic links pointing outside of the chroot environment do not work, hard links will work because they reference specific inodes

- It is possible more isolation than a chroot environment is necessary
- Modern technologies allow isolation of the entire system as opposed to simply the file system:
  - Virtualization creates entire systems running on top of the kernel
  - Containerization is a means of creating sparse containers having only what is required to run a particular piece of software

Securing Grub

- The Concept:
  - Grub is capable of preventing unauthorized users from changing boot parameters

    Access to these parameters may allow a user to compromise a system
  - Securing Grub 1:

    Grub 1 does not support unique user accounts, only passwords for entries

    Specify `password agoodpw` in menu entry in `grub.conf` in order to password protect
  - Securing Grub 2:

    Grub 2 allows for a two types of password

    A global password that protects editing menu entries

    An operating system password that protects booting specific menu entries.

To configure Grub 2 users:

Edit `/etc/grub.d/01_users`

```
set superusers="bob"
password bob somepw
password john anotherpw
```

Allow a user to edit a menu entry:

Edit `/etc/grub.d/40_custom`

```
  menuentry ... --users john {
    set root=(hd0,msdos1)
    linux /vmlinuz-*
  }
```

Menu entry samples may be gathered using `grep menuentry /boot/grub2/grub.cfg`

Use `grub2-mkconfig -o /boot/grub2/grub.cfg` to rebuild grub configuration

326.2 Host Intrusion Detection

Threat Detection

- Using AIDE for Threat Detection
  - The Concept:

    AIDE stands for Advanced Intrusion Detection Environment

    AIDE maintains a database of file signatures:

      Verifies file signatures periodically (when configured)

- Can detect changes in file properties or content
- It is important to keep a copy of the database on offline media for maximum security
- AIDE is most useful for monitoring static operating system files

Configuring and using AIDE:

- Primary configuration file `/etc/aide.conf`
  - Well documented
  - Simple syntax for defining what to monitor:

    From aide.conf:

```
\#p:      permissions
\#i:      inode:
\#n:      number of links
\#u:      user
\#g:      group
\#s:      size
\#b:      block count
\#m:      mtime
\#a:      atime
\#c:      ctime
\#S:      check for growing size
\#acl:           Access Control Lists
\#selinux        SELinux security context
\#xattrs:        Extended file attributes
\#md5:    md5 checksum
\#sha1:   sha1 checksum
\#sha256:        sha256 checksum
\#sha512:        sha512 checksum
\#rmd160: rmd160 checksum
```

```
\#tiger:  tiger checksum

\#haval:  haval checksum (MHASH only)
\#gost:   gost checksum (MHASH only)
\#crc32:  crc32 checksum (MHASH only)
\#whirlpool:     whirlpool checksum (MHASH only)

FIPSR = p+i+n+u+g+s+m+c+acl+selinux+xattrs+sha256
\#R:             p+i+n+u+g+s+m+c+acl+selinux+xattrs+md5
\#L:             p+i+n+u+g+acl+selinux+xattrs
\#E:             Empty group
\#>:             Growing logfile   p+u+g+i+n+S+acl+selinux+xattrs
\# You can create custom rules like this.
\# With MHASH...
\# ALLXTRAHASHES = \
sha1+rmd160+sha256+sha512+whirlpool+tiger+haval+gost+crc32
ALLXTRAHASHES = sha1+rmd160+sha256+sha512+tiger
```

May set an alias for simplified use

Know how to write monitoring rules

Commands:

`aide --init` establishes original database

`aide --check` compare current file system to database

- OpenSCAP for Compliance

  SCAP stands for Security Content Automation Protocol

  SCAP provides for both Vulnerability Assessment and Security Compliance

  OpenSCAP is an implementation of the protocol that can audit your system for a provided template

Packages:

Debian: `libopenscap8`

Red Hat: `openscap-scanner`

- Linux Malware Detect (LMD or maldet) for Threat Detection

The Concept:

LMD uses external signatures to check for threats

LMD can be configured to scan a system regularly

Pre-requisites for real-time notification:

Install package `inotify-tools`

You may choose to install ClamAV which LMD integrates with for improved scanning performance

Installation:

Download from tarball at http://www.rfxn.com/downloads/maldetect-current.tar.gz

Untar and run `install.sh` with `root`

Command `maldet`

`-a`: Will initiate an immediate full scan. A specific path to scan may also be specified.

`-e <report>`: Will display a report generated from execution.

`-m`: Will start monitor mode.

From the docs:

```
-m, --monitor USERS|PATHS|FILE|RELOAD
 Run maldet with inotify kernel level file create/modify monitoring
 If USERS is specified, monitor user homedirs for UID's > 500
 If FILE is specified, paths will be extracted from file, line spaced
 If PATHS are specified, must be comma spaced list, NO WILDCARDS!
 e.g: maldet --monitor users
```

```
e.g: maldet --monitor /root/monitor_paths
e.g: maldet --monitor /home/mike,/home/ashton
```

Run `maldet --help` for documentation.

LMD may be started in monitor mode using built in systemd targets as well

Main configuration file is in `/usr/local/maldet/conf.maldet`
`/usr/local/maldet/monitor_paths` contains paths that maldet should watch

Important configurations:

- `email_alert`
  - `0` to disable email alerts
  - `1` to enable email alerts
- `email_addr`
  - Email address to send reports to
- `quarantine_hits`
  - `0` does not quarantine on detection
  - `1` does quarantine on detection
- `autoupdate_signatures`
  - `0` disables automatic signature updates
  - `1` enables automatic signature updates
- `autoupdate_version`
  - `0` disables automatic LMD updates
  - `1` enables automatic LMD updates
- `scan_clamscan`
  - If `clamav` is installed, `maldet` will use `clamav` scanning for improved performance

- `0` disables clamav integration.
- `1` enables clamav integration.

- Rootkit Detection
  - AIDE can track static binaries for changes
  - Kernel modules are common targets for rootkits as they are loaded with root access
  - When using tools that check for rootkits, it is important you can trust the binaries those tools may reference
  - Working with `chkrootkit`
    - Install `chkrootkit` package on Debian based systems
    - Must be compiled from source on Red Hat based systems
    - `sudo chkrootkit` will check for rootkits
    - `sudo chkrootkit -q` will check for rootkits and have less verbose output
    - Configuration file located at `/etc/chkconfig.conf`
  - Working with `rkhunter`
    - Use package `rkhunter`
      - Note: Must use an EPEL repository for Red Hat based distros
  - `rkhunter`
    - `rkhunter --update` is used to update `rkhunter` signatures.
    - `rkhunter -c` check for rootkits:
      - Generates output log in `/var/log/rkhunter/rkhunter.log`
      - `--cronjob` stops periodic prompts for input
      - `-rwo` reports warnings only

System Auditing

- A brief word on logging
  - service logs (depends on service config)
  - syslog (typically written in `/var/log`)
  - systemd-journald (uses syslog by default)
- Auditing goes deeper than logs
  - `auditd` process handles auditing
  - Writes to `/var/log/audit/audit.log`
  - Contains the following types of information:

    Type of audit

    Timestamp (in EPOC)

    PID

    UID (Effective UID)

    Audit UID (original UID - who started process, think setuid)

    `Sess`: session id (important for even correlation)

    `subj`: selinux info

    `msg`: message

    Name of the executable

    Binary
  - Primary `auditd` configuration file: `/etc/audit/auditd.conf`
  - Can create rules in `/etc/audit/rules.d` or `/etc/audit/audit.rules`

- Sending `auditd` messages to remote systems
  - Install `audispd-plugins` package
  - To send to `auditd` on another server:
    - Configure client system
      - Edit `/etc/audisp/plugins.d/au_remote.conf`
        - Set `active=yes`
      - Edit `/etc/audisp/audisp-remote.conf`
        - Set `remote_server = target-server-hostname`
      - Configure `auditd` server
        - Edit `/etc/audi/auditd.conf`:
          - Set `tcp_listen_port = <port_num>`
          - Note there is not a default port
        - Set SELinux port context for chosen port
        - Restart `auditd`: requires reboot
    - Sending `auditd` messages to remote `syslogd`
      - Edit `/etc/audisp/plugins.d/syslog.conf`
        - Set `active=yes`
  - Audit Utilities
    - `ausearch` — search for specific events:
      - `ausearch -i -a <event_code>`
      - `-i` Interpret
      - `-a` traces events by event code
      - `ausearch --help`

- Filtering options such as `uid` or `pid`
- `aureport` — Generic reporting utility, get details for events
  - Provides summary when ran with no options
  - `aureport --help`
  - `-a` — SELinux Events
  - `-l` — Login information
  - `-ma` — Manditory access control
- `autrace` — creates an audit trace for a specific command issued
  - `autrace <command_with_full_path> <command options>`
  - Produces an `aureport` command that may be used to view audit records with the executed command

Custom Audit Rules

- `auditctl` allows you to create custom rules
  - `-w` — Watch
  - `-p` — Properties
  - `-k` — Key for audit log
  - `-a exit,always ->` — Trigger audit at end of `systemcall`
  - `-F` — Filter
  - `-C` — Comparison
  - `-l` — Print current rule list:
    - Can take output from this command to make persistent rules in `/etc/audit/audit.rules`
  - `-D` — Delete rules

- View man page for more detailed information and examples
- Audit Rules apply FIFO; the earliest rule always apply

Predefined audit sets

- Look in `/usr/share/doc/audit-<version>/`
- Review rules sets available sets
- Simply copy conf files of preferred rule sets into `auditd.rules`
- Managed audit file size:
  - Copy `auditd.cron` from docs into `crontab`
- Keystroke Logging may be achieved using a PAM module
  - Edit `/etc/pam.d/system-auth`
  - `session required pam_tty_audit.so enable=root`
  - View output of module with `aureport --tty`

326.3 User Management and Authentication

Linux Login Essentials

- `login.defs`
  - The login process originally relied on `/etc/login.defs` entirely for authentication
  - Over time, much of the functionality of this file was replaced by PAM modules and similar services
- The `chage` command may be used to manage password expiration for users who have an entry in `/etc/shadow`
  - Only `root` may successfully run `chage` in all cases except for the `-l` flag which returns when a users password is due to expire
  - `chage` is currently unaffected by `/etc/login.defs`

- Notable switches:

  `-E <YYYY-MM-DD> LOGIN` — Expire the password for `LOGIN` on YYYY-MM-DD; use a value of -1 to remove an expiration date

  `-I <days> LOGIN` — Number of days of user inactivity to expire LOGIN

  `-W <days> LOGIN` — Number of days in advance to beginning warning LOGIN of password expiration

  Most options may be made inactive by supplying a value of -1

  Ran without options, `chage` is interactive

- NSS Concepts and Configuration
  - The Concept:

    The Name Service Switch, or NSS, is responsible for connecting calls for information from a system database to a back-end service

    Some important databases supported by NSS include:

    `group`: Groups of users

    `netgroup`: Network-wide list of host and users, used for access rules

    `hosts`: Host name and numbers

    `service`: System services

    `passwd`: User passwords

    `shadow`: Shadow user passwords

    Services include (but are not limited to):

    `files`: Local system configuration files (ie `/etc/passwd`)

    `nis`: Network Information Service

    `sss`: System Security Service Daemon (only valid for `passwd`, `group`, `services`, and `netgroup`)

    Certain databases may support certain services depending on the service

Hosts may use the `dns` service

The standard installation of most distributions use the files service by default

- Working with NSS

  Changes may be required to `/etc/nsswitch.conf` if non-default databases are to be used

  Typically, this is required when a system will use LDAP for system login and user information

  The entries in `/etc/nsswitch.conf` are read in on startup by most programs and will not be re-read

  A snippet of a typical `nsswitch.conf` file:

```
passwd:     files sss
shadow:     files sss
group:      files sss
hosts:      files dns
```

  Databases attempt services in order from left to right

  In the snippet, all listed databases will look at local system files first

  If no match is found, the next service will be queried

- See `man nsswitch.conf` or the comments in `/etc/nsswitch.conf` for more detail

PAM Concepts

- The Concept:
  - PAM stands for Pluggable Authentication Module.
  - PAM has worked its way into the core login process of modern Linux distributions.
  - PAM is made up of a number of modules, each handling a unique function involved in portions of authentication and authorization processes.

- The standard Linux login process works through a hierarchy of authentication and authorization checks using various modules.

- Configuring PAM:
    - Modules are engaged within files that vary by distribution in `/etc/pam.d`.
    - They are provided in the format `<realm> <control> <module> <arguments>`.
    - PAM modules are divided into four management groups or realms:

        auth: Modules in this group are used for authentication purposes.

        account: This group takes care of account related work such as verifying account details.

        password: This group handles password related work.

        session: These modules interact with user sessions.
    - A module may have functionality in more than one management group.
    - Modules are stacked, they run in a set order and, based on specified controls, a certain sequence of modules must pass to approve a request.
    - Modules may have varying degrees of control specified by certain keywords.

        required: A required modules must succeed for stack success however a required module failure will not stop remaining checks from being invoked though the stack will still fail.

        requisite: If a requisite module fails, the stack fails immediately - no further modules are invoked.

        sufficient: If any sufficient module in a stack succeeds, then remaining sufficient modules are not invoked.

        optional: Optional Modules only causes failure if it is the only module in the stack.
    - Most modules accept varying degrees of arguments that may be found within the module's documentation.
    - Programs that require account services are able to hook a PAM module to satisfy any requirements.

- Notable Modules:
    - Password complexity may be enforced using `pam_cracklib.so`.

        This module performs basic complexity checks for users setting a password.

Capabilities include:

- `minlen=N` — Password must be N characters or more.
- `retry=N` — A user may only be prompted N times before the module returns an error.
- `enforce_for_root` — Root must also follow set rules (off by default).
- `credit parameters` — If above 0, provide a reduction up to N in password minlength for each occurrence of a given class.
    - `dcredit=N` — Reduce minlength up to N for each digit in the provided password.
    - `ucredit=N` — Reduce minlength up to N for each uppercase letter in the provided password.
    - `lcredit=N` — Reduce minlength up to N for each lowercase letter in the provided password.
    - `ocredit=N` — Reduce minlength up to N for each symbolic character the in provided password.
- Some distributions are beginning to favor `pam_pwqaulity.so` instead (see `man pam_pwquality`)

Example: `password requisite pam_cracklib.so retry=3 minlength=8`

- This line in the PAM stack will allow a user to submit up to 3 passwords that must satisfy a minimum length of 8 characters:
    - See `man pam_cracklib` for more detail.

The `pam_tally.so` and `pam_tally2.so` are able to count login attempts by a user and to act on a number of consecutive failed logins.

`pam_tally.so` originally served this purpose.

- Logs attempts to `/var/log/faillog` by default.
- May use `faillog` to view a user's last failed login attempt.
    - Deprecated in favor of `pam_tally2.so`
    - See `man pam_tally`

Regarding `pam_tally2.so`

Logs to `/var/log/tallylog`

Important auth parameters

`deny=n` will instruct tally to deny login after n failed login attempts.

`unlock_time=n` will allow denied users to attempt again after `n` seconds.

`even_deny_root` will make root subject to lock out as well (this is not default).

Example implementation:

```
auth required pam_tally2.so deny=4 even_deny_root unlock_time=1200
```

Locks accounts (including root) for 20 minutes after 4 failed login attempts.

Kerberos

- Kerberos and local domains.
- The Concept:
  - Kerberos enhances security by providing network entity authentication in a way that does not require sending unencrypted passwords over the wire.
  - This is achieved through symmetric key cryptography and use of a ticketing system.
  - Kerberos terms:

    KDC: Key Distribution Center, a trusted third party used to issue tickets for principals to services.

    Realm: The computers managed by a KDC and any secondary KDCs.

    Principal: A Kerberos user identity.

    Ticket: A set of credentials

    TGT: Ticket granting ticket, this is used as a means of identifying a principal in Kerberos.

TGS: Ticket granting server, this is used to grant access to particular services provided a principal has a valid TGT.

- The Process:

  A user authenticates to a KDC by sending principle identity and credentials.

  So long as the KDC is able to authenticate the principal, it will issue a TGT and sends it to the user (The TGT will be encrypted by the user's key).

  The TGT is locally decrypted using the user's key which is derived by using the user's password.

  The ticket is stored locally in a credential cache which may be accessed by other Kerberos-aware services.

  The user's principal and key are stored locally in a `keytab` file after authentication has been performed which prevents unnecessary callbacks to the KDC.

  The TGT is good for a finite period.

  As long as the TGT has not expired, the user will not be prompted to use password authentication.

  Each time the user uses a network service, the service will transparently validate the TGT by retrieving a service ticket from the TGS.

- Important notes:

  Kerberos requires exact time synchronization within the realm to work correctly.

  Kerberos is dependent on accurate domain name resolution to work correctly.

  DNS domains generally have a one to one relationship with Kerberos Realms.

- Kerberos configuration is managed in `/etc/krb5.conf`

  - Example file:

```
 [libdefaults]
default_realm = ATHENA.MIT.EDU
dns_lookup_kdc = true
dns_lookup_realm = false
```

```
[realms]
  ATHENA.MIT.EDU = {
    kdc = kerberos.mit.edu
    kdc = kerberos-1.mit.edu
    kdc = kerberos-2.mit.edu
    admin_server = kerberos.mit.edu
    master_kdc = kerberos.mit.edu
  }
  EXAMPLE.COM = {
    kdc = kerberos.example.com
    kdc = kerberos-1.example.com
    admin_server = kerberos.example.com
  }

  [domain_realm]
    mit.edu = ATHENA.MIT.EDU
```

- The file is broken up into a few major sections:

  `libdefaults` — Settings used by the Kerberos V5 library.

  `realms` — Realm-specific contact information and settings.

  `domain_realm` — Maps server hostnames to Kerberos realms.

  `capaths` — Used for cross-realm authentication.

  `appdefaults` — Application specific configuration.

  `plugins` — Plugin configuration information.

- Working with Kerberos tickets:
  - The `kinit` command will reach out to the KDC for a TGT.

    Once the TGT has been established, the user will not be prompted for a password when working with Kerberos-enabled services.

The TGT will eventually expire or will be removed when the user ends their current session.

- The `klist` command will show a user's current tickets on a given system.
- The `kdestroy` command may be used to destroy an active TGT.

Understanding SSSD

- Configure and use SSSD
  - The Concept:

    SSSD is short for the System Security Services Daemon

    SSSD provides a set of daemons to manage access to remote directories and authentication mechanisms

    Key integrations include:

      Active Directory

      FreeIPA

      LDAP

      Kerberos

    SSSD may also be configured for a local domain
  - The primary configuration file for SSSD is `/etc/sssd/sssd.conf`

    Example configuration located in `/usr/share/doc/sssd-common-<VERSION>/sssd-example.conf`

    The file must have mode 0600 in order to be picked up by SSSD

    The file uses an INI format style with sections in brackets followed by key-value pairs for configuration

    Notable directives:

      `id_provider` — What backend to use for identity

      `auth_provider` — What backend to use for authentication

        Providers include Active Directory, IPA, LDAP, Kerberos, and local

        Each type will be covered briefly below

    See `man sssd.conf` for more information

- Configurations for Active Directory

    SSSD may use Active Directory 2008 R2 or later for directory support

    The AD provider enables the LDAP and Kerberos v5 with optimizations for Active Directory environments

    Notable directives

        `ldap_id_mapping = True` — Maps UID and GID values using the objectSID parameters in AD

        `ldap_schema = ad` — Specifies the LDAP schema for use with AD

        `ad_server, ad_backup_server` — Comma-separated list of hostnames of AD servers to use in order of preferences

        `ad_hostname` — The local hostname as AD knows it

    See `man sssd-ad` for more details

- Configurations for FreeIPA

    This has very similar features to the Active Directory provider except it is optimized to work with IPA

    Notable directives:

        `ipa_server`, `ipa_backup_server` — Comma-separated list of hostnames of IPA servers to use in order of preferences.

        `ipa_hostname` — The local hostname as IPA knows it.

    See `man sssd-ipa` for more detail.

- Configurations for LDAP

    Notable directives:

        `ldap_uri = ldap://ldap.mydomain.org` — Directory url to connect to `ldap-search_base - dc=mydomain,dc=org` — LDAP search base to use for lookups

`ldap_tls_reqcert = demand` — Require encrypted connection

`cache_credentails = true` — Reduce login time and load on ldap server by caching lookups

See `man sssd-ldap` for more detail.

- Configurations for Kerberos

  Kerberos only provides auth and must be paired with an identity backend, such as LDAP

  Kerberos relies on `.k5login` for access to a server

  `krb5_server = 192.168.1.1` — Kerberos Server to be used by SSSD

  `krb5_realm = EXAMPLE.COM` — Kerberos Realm

  See `man sssd-krb5` for more detail

  Configurations for Local Domains

  SSSD will use its native database

  `id_proivder` and `auth_provider` must be set to `local` to use the local domain

  See `man sssd.conf` for more detail

- Configure NSS and PAM for use with SSSD
  - SSSD may be used in NSS by specifying the SSS service in `/etc/nsswitch.conf`

    Note the SSS service is only supported for the following databases:

    `passwd`

    `group`

    `services`

    `netgroup`

        - There are some NSS specific configurations in `/etc/sssd.conf`
            - It is possible to only allow certain NSS users or groups using the `filter_users` and `filter_groups` directives
            - It is also possible to override some settings:
                - `override_homedir` may be used to override a users home directory
                - `override_shell` may be used to override the login shell for all users
                - `allowed_shells` may be used to restrict which shells may be used
    - SSSD integrates with pam using the `pam_sss.so` module
        - Notable Parameters:
            - `forward_pass` — Allow other modules to use the password supplied to `pam_sss`
            - `domains` — A comma-separated list of sssd domains to allow authentication against
            - There are some PAM specific configurations in `/etc/sssd.conf` which mostly center around timeout values
- There are a number of utilities provided by the `sssd-tools` package
    - The following commands allow for local user manipulation using sssd:
        - `sss_useradd`
        - `sss_userdel`
        - `sss_usermod`
    - There are similar commands to manage groups as well: `sss_groupadd`, etc.
    - These commands behave similarly to their non-sss counterparts in terms of options
    - The `sss_obfuscate` command will provide weak hashing for clear text passwords to use in sssd configuration files

- The `sss_cache` invalidates records in SSSD cache. Invalidated records are forced to be reloaded from the server as soon as related SSSD back-end is online:
    - `-u`: Invalidates the cache for a given user
    - `-g`: Invalidates the cache for a given group
    - `-s`: Invalidates the cache for a given service
    - Using `-U`, `-G`, or `-S` will invalidate records for all users, groups, or services, respectively

326.4 FreeIPA Installation and Samba Integration

Overview of FreeIPA

- FreeIPA is a suite of tools that are similar (though not matching) to Microsoft's Active Directory
- FreeIPA provides identity and authentication services
- FreeIPA can store attributes such as:
    - User sudo rules for central sudo management
    - Autofs configuration
    - SELinux user mappings
- Several components make up FreeIPA
    - 389 Directory Server: Directory server used for Identity Service.
    - MIT Kerberos: Kerberos server and KDC used for Authentication Service.
    - DNS (BIND): Server for Domain Name System service.
    - NTP: Time synchronization services.
    - Dogtag: PKI Certificate Management Service.

- Prerequisites for installing FreeIPA
  - A FreeIPA server must have a static hostname configured in `/etc/hosts`
  - DNS checks that must succeed:
    - The DNS domain of an IPA host may not be changed after IPA is configured
    - The hostname cannot be *localhost* or *localhost6*
    - The hostname must be fully-qualified (*ipa.example.com*)
    - The hostname must be resolvable
    - The reverse of address that it resolves to must match the hostname
  - Firewall ports must be available:
    - Kerberos tcp/88 TCP/464 UDP/88 UDP/464
    - HTTP/S TCP/80 TCP/443
    - DNS UDP/53
    - NTP UDP/123
    - LDAP/S TCP/389 TCP/636
  - About `certmonger`
    - The Concept:
      - The `certmonger` daemon monitors certificates for impending expiration
      - It can optionally refresh soon-to-be-expired certificates with the help of a CA
      - It can drive the entire enrollment process from key generation through enrollment and refresh
  - Working with `certmonger`:
    - `getcert list` shows all certificates being tracked by `certmonger`
    - `ipa-getcert list` shows all IPA-issued certificates

`ipa-getcert request -f /path/to/server.crt -k /path/to/private.key -r`

Uses IPA CA configured in `/etc/ipa/default.conf`

`-f` is for certificate file.

`-k` is for private key file.

`-r` have `certmonger` attempt to automatically renew this certificate when it nears expiration.

Installing and Configuring FreeIPA

- Installing FreeIPA
    - FreeIPA is provided by the package `ipa-server`
    - If you desire to run integrated DNS locally, more packages may be required
    - Initiate install process with `ipa-server-install`

      The command prompts for various configuration settings including:

      Domain name

      Kerberos realm name

      Required passwords

      DNS forwarder configuration (if desired)
    - Configuring Replica nodes:

      `ipa-replica-install`

      Configures a server to be a replica of an existing domain

      May use a replica file generated by `ipa-replica-prepare`

      May also be configured by hand using switches

- `ipa-replica-prepare`
  - Generates a replica file when ran on an already configured master server
  - Requires accurate DNS and reverse lookup host information
  - Must provide FQDN of the target server (the one you plan to make a replica)
- `ipa-replica-manage`
  - Manages replication agreements between IPA servers
  - Notable subcommands:
    - `list` shows all servers a given server replicates with
    - `connect` initiates a replication agreement to a target server
    - `disconnect` terminates a replication agreement with a target server
    - `del` removes all replication agreements AND data from a target server
    - `re-initialize` forces a full re-initialization of the server, retrieving data from the server specified with the `--from` option
    - `force-sync` immediately flushes any data to be replicated from a server provided with the `--from` option

• Active Directory replication and Kerberos cross-realm trusts with IPA
- Requires the package `ipa-server-trust-ad`
- Prerequisites
  - Install the `ipa-server-trust-ad` package
  - IPv6 must be enabled on the IPA server!
  - Both IPA and AD hosts must be able to resolve each other's DNS names
  - IPA server must be accessible over the following ports:
    - TCP ports: 80, 88, 443, 389, 636, 88, 464, 53, 135, 138, 139, 445, 1024-1300

UDP ports: 88, 464, 53, 123, 138, 139, 389, 445

The `ipa-adtrust-install` utility must be ran to to prepare the IPA server for trust

The `ipa trust-add` command creates the trust agreement

It assumes a one-way trust to AD by default

Note you must have either domain administrator credentials on the AD domain or a trust-secret configured to perform this

The option `--two-way=true` will configure a two-way trust

See `ipa help trust-add` for more information

Working with FreeIPA

- `ipa-client-install`
  - Join a machine to a Free IPA domain using an existing domain user.
  - If installed, SSSD is automatically configured.
  - If SSSD is not installed, NSS LDAP and appropriate pam modules are configured.
  - Kerberos and NTP clients are also configured on the local system.
  - In package `ipa-client`.
- The `ipa` command is the general command line utility used to interact with IPA.
  - The command has a number of subcommands for various tasks:

    `user-add user1 --first foo --last bar`— Create the user `user1` with the first name of `foo` and last name of `bar`.

    `group-add group1 --desc "The description for this group"` — Create the group `group1` with the description "The description for this group".

    `group-add-member group1 --users=user1` — Add `user1` to `group1`.

  `user-find token` — Find users with attributes matching 'token'

  `help` — Provides topic help or subcomand help; this must be run on a system which has had IPA configured.

- See `man ipa` for additional details.

Topic 327 Access Control

327.1 Discretionary Access Control

Basic System Permissions and Extended Attributes

- Understand and manage file ownership and permissions, including SUID and SGID
    - File ownership is controlled with `chown` which may only be run by `root`
    - Permission mode is controlled using the `chmod` command and uses an octal notation

        The read bit is represented with *4* or symbolically as *r*

        The write bit is represented with *2* or symbolically as *w*

        The execute bit is represented with *1* or symbolically as *x*

        Permissions are assigned by 4 bits:

            A special permission bit used by SUID and SGID (covered later)

            File owner

            File group owner

            World
    - SUID Permission

        SUID is a special permission that allows the file being executed to be executed with the privileges of the owner

        SUID is numerically represented by 4 in the special permission bit

SUID is symbolically represented on the owner permission bit by *S* (uppercase) when the file is not executable and *s* (lowercase) when the file is executable

Note that modern kernels tend to restrict which files may actually use this permission to enhance security

- SGID Permission

  SGID is represented symbolically by *s* in the group permission bit or a *2* in the special permission bit

  When SGID is set on a file, the file will execute under the authority of the group owner of the file

  When SGID is set on a directory, each file created in the directory will inherit the directory's group ownership

- Sticky Bit

  The sticky bit is represented symbolically by a *t* in the world execute permission bit

  The sticky bit is numerically represented by *1* in the special permission bit

  The sticky bit makes a file only deletable by the owner of the file

  It is most prominently used by the `/tmp` directory

- Extended File Attributes

  - The Concept:

    In Linux, the `ext2`, `ext3`, `ext4`, `JFS`, `ReiserFS` and `XFS` filesystems support extended attributes (abbreviated `xattr`) if the `libattr` feature is enabled in the kernel configuration.

    Any regular file may have a list of extended attributes denoted by the name

    The name must:

      Be a null-terminated string

      Be prefixed by a namespace identifier and a dot character

    Currently, four namespaces exist:

      `user`: A general purpose namespace with no restrictions regarding naming or contents.

`trusted`: Attributes in this class are used to implement mechanisms in user space which keep information in extended attributes to which ordinary processes should not have access.

`security`: Used by SELinux.

`system`: Primarily used by the kernel for access control lists.

Extended attributes are not widely used in user-space programs in Linux, although they are supported in the 2.6 and later versions of the kernel

- Managing extended attributes

  The `getfattr` — For each file in a provided path, `getfattr` displays the file name, and the set of extended attribute names (and optionally values) which are associated with that file

  Notable options:

  `-n <name>` returns the named attributes for a given path

  `-m <pattern>` returns attributes with attributes matching the provided pattern

  The `setfattr` command associates a new value with an extended attribute name for each specified file

  Notable options

  `-n <name>` returns the named attributes for a given path

  `-v <value>` the new value to assign to a given attribute

  `x <name>` remove an attribute entirely

Using ACLs

- Understand and manage access control lists
  - The Concept:

    ACL stands for Access Control Lists

    ACLs allow for more fine-grained discretionary access rights for files and directories

There are four entry tags that ACL permissions may be assigned to:

- user: The file owner or a specified user; may be abbreviated `u`
- group: The group owner or a specified group; may be abbreviated `g`
- mask: An entry that specifies that maximum access which can be granted by any ACL entry EXCEPT the user entry for the file owner; may be abbreviated as `m`
- other: An entry that specifies access granted to any process that does not match any user or group ACL entries; may be abbreviated `o`

Read, Write, and Execute permissions may be assigned to any specified identifier

ACLS are supplied in the following format: `entry_tag:identifier:object_permissions`

- Examples:
  - The user john has read and write access: `u:john:rw-`
  - The group staff has read access: `g:staff:r--`
  - Other access is no access: `o::---`

Order of permission application

- The file owner permission applies above all other entries
- User permissions override group permissions up to the level allowed by a set ACL mask
- Group permissions up to the level allowed by the ACL mask
- Other ACL permissions apply for anything not matched by a user or group

• Managing ACLs:

`getfacl`:

- get file access control lists For each file
- `getfacl` displays the file name, owner, the group, and the Access Control List (ACL)
- If a directory has a default ACL, `getfacl` also displays the default ACL

Non-directories cannot have default ACLs.

If `getfacl` is used on a file system that does not support ACLs, `getfacl` displays the access permissions defined by the traditional file mode permission bits.

`setfacl`

Set file access control lists

Notable Options:

- `m` modify
- `x` remove

Examples:

Granting an additional user read access:

```
setfacl -m u:lisa:r file
```

Revoking write access from all groups and all named users (using the effective rights mask):

```
setfacl -m m::rx file
```

Removing a named group entry from a file's ACL:

```
setfacl -x g:staff file
```

Copying the ACL of one file to another:

```
getfacl file1 | setfacl --set-file=- file2
```

Copying the access ACL into the Default ACL:

```
getfacl --access dir | setfacl -d -M- dir
```

327.2 Mandatory Access Control

Understanding Access Control

- Access control comes in two primary forms:
  - Mandatory Access Control (MAC):

    MAC is the security style provided through systems such as SELinux and AppArmor.

    Access is controlled through context rather than by the owner.

    Each system resource has a type associated with it, and the kernel will only let users who have access to the given type to access the resource.

    This is known as Type Enforcement (TE).

    MAC is further enforced through Role assignment, also known as Roles Based Access Control (RBAC)
  - Discretionary Access Control (DAC):

    The owner controls permission.

    Traditional POSIX permissions are an example of DAC.

SELinux

- The Concept:
  - SELinux is a Mandatory Access Control system developed by the NSA and hooked into many Linux distributions as a Kernel module.
  - SELinux adds another layer of security in the Linux Kernel.
  - SELinux uses a set of defined policies that assign roles and contexts to users and system objects.
  - Roles must be permitted to access to a context in order to use resources confined in that context; this applies no matter the user (root included).

- SELinux can operate in two primary modes:

  Enforcing: Access violations are denied.

  Permissive: Access violations are allowed but logged.

  These modes may be set in `/etc/selinux/config` or by using the `setenforce` command.

- Configuring SELinux:
  - Configuration files located in `/etc/selinux/*`
  - Enabling and disabling SELinux:

    `getenforce` determines what the current SELinux enforcement level is.

    `setenforce` allows for changing the running SELinux configuration in a non-persitent way.

    `selinuxenabled` is optimized for use with shell scripts and returns the system's SELinux status.
  - There are a number of SELinux security settings control through booleans.
  - The booleans may be managed through a few key commands:
  - `getsebool`, `setsebool`, `togglesebool`
  - Tools for managing SELinux contexts on files:

    `restorecon` is used to restore the default security context to a given file.

    `fixfiles` is a wrapper around `restorecon` that adds some simplified functionality such as a verify function and a relabel function.

    `setfiles` may be used to set SELinux file security contexts.
  - Tools for testing SELinux contexts and roles:

    `newrole` — Run a shell with a new SELinux Role

    `runcon` — Run a command in a given SELinux Context.
  - General purpose SELinux management commands:

    `semanage` consists of several subcommands that may be used to configure SELinux in general.

`sestatus` provides general SELinux status information including the policy that has been loaded as well as the enforcing status.

`seinfo` allows for a way to query parts of a provided SELinux policy.

`apol` — A GUI SELinux policy analysis tool that has similar functionality to `seinfo` except in GUI form.

- SELinux auditing tools:

  `sealert` — A tool for scanning avc log entries and generating reporting information.

  `seaudit` — A graphical tool for viewing logs and filtering based on certain SELinux Policies.

  `seaudit-report` — A deprecated tool used to generate reports based on a given SELinux Policy and log file.

  `audit2allow` — Generate SELinux policy to `allow/dontaudit` rules from logs of denied operations.

  `audit2why` — Translates SELinux audit messages into a description of why the access was denied (`audit2allow -w`).

MAC Alternatives

- AppArmor
  - AppArmor is an alternative to SELinux made popular with Ubuntu.
  - It is known for being less cumbersome to manage than SELinux.
  - AppArmor works by assigning types to file paths rather than inode (as SELinux does).
  - `apparmor` is the system daemon that managed AppArmor
  - AppArmor has two modes it can operate in: Enforcement or Complain:

    These modes are comparable to SELinux Enforcing and Permissive.
  - Policies can be developed using the `aa-genprof` and `aa-logprof` commands.

- Smack
    - Smack is another MAC option available.
    - It must be compiled into the kernel.
    - Smack uses extended file attributes for label assignment.
    - Much like SELinux, many commands will provide label details on a Smack system using the -Z flag.
    - The `chsmack` command may also be used to query and set label information.

327.3 Network File Systems

NFSv4 Improvements

- Understand NFSv4 security issues and improvements
    - Firewall Issues:

        NFSv3 relied on portmapper to dynamically assign server ports at runtime which made running NFS behind a firewall challenging

        The random firewall ports were fixed by TCP port 2049 in NFSv4

        May still use dynamic ports if required
    - Access Security:

        NFSv3 originally relied on NIS for handling user-level security and mapping

        All hosts were members of the same domain so that NFSv3 security was based on the host instead of the user

        NIS no longer sees mainstream use

        The host-based security has been enhanced with Kerberized authentication in NFSv4

            Kerberos support requires the use of the GSS (Generic Security Services) API

            The configuration and implementation of GSSAPI may vary by distribution

GSSAPI implements public key authentication methods such as LIPKEY and SPKM

- Pseudo File System:

  Several exports may be mounted at once by mounting their parent directory

  Example:

  Assuming the exports `/data/secure`, `/data/files`, and `/data/etc` existed and were each exported on `nfs.example.com`

  The command `mount nfs.example.com:/data /mnt` would mount all of the exports to `/mnt`

NSF in practice

- The package `nfs-utils` must be installed to run an NFS server.
- The daemon `nfs` provides service.
- Exports are configured using the file `/etc/exports`:
  - Each line contains an export point and a whitespace-separated list of clients allowed to mount the file system at that point.
  - Each listed client may be immediately followed by a parenthesized, comma-separated list of export options for that client.
  - No whitespace is permitted between a client and its options list.
  - Each line may have one or more specifications for default options after the path name, in the form of a dash ("-") followed by an option list.

    The options list is used for all subsequent exports on that line only.
  - Clients may be provided as:

    Hostname

    Host IP addresses

    Networks

Wildcard entries

- To apply changes to this file, run `exportfs -ra` or restart the NFS server.
- Example:

```
/               master(rw) trusty(rw,no_root_squash)
/projects       proj*.local.domain(rw)
/home/joe       pc001(rw,all_squash,anonuid=150,anongid=100)
/pub            *(ro,insecure,all_squash)
```

- Notable export configuration options:

  `no_root_squash` — Root on the client system will map to root on the NFS server.

  `rw` — Allows read and write operations on a given export (exports are read-only by default).

  `sec=` — Takes a colon-delimited list of security flavors and restricts the export to clients using those flavors.

    The order of flavors provided matters (earlier entries are preferred).

    Available security flavors include:

      `sys` (the default--no crypto-graphic security)

      `krb5` (authentication only)

      `krb5i` (integrity protection)

      `krb5p` (privacy protection)

- Configure `/etc/idmapd.conf`

  The id mapper daemon is required on both client and server.

  It maps NFSv4 `username@domain` user strings back and forth into numeric UIDs and GIDs when necessary.

  The client and server must have matching domains in this configuration file:

```
[General]
```

```
        Verbosity = 0
        Pipefs-Directory = /var/lib/nfs/rpc_pipefs
        Domain = vanemery.com

        [Mapping]

        Nobody-User = nfsnobody
        Nobody-Group = nfsnobody
```

- Configure NFSv4 client

  Clients must have `nfs-utils` installed which will allow the use of the `mount` command to mount NFS volumes.

  The command `showmount -e nfs.example.com` list the available exports on `nfs.example.com` from the perspective of your host.

  If Kerberos authentication is to be used, you must set a security option mount option `sec=` with the appropriate authentication flavor supported by the NFS server.

Understand and use NFSv4 ACLs

- The Concept:
  - NFSv4 introduces access control lists (ACLs) on files shared over NFS.
  - The ACLs allow for enhanced integration for the Windows OS.
  - The ACLs also support similar functionality to typical Linux ACLs.
- Working with NFS ACLs:
  - Provided by the `nfsv4_acl_tools` package.
  - The key commands for managing NFS ACLs are `nfs4_setfacl` and `nfs4_getfacl`.

- An NFSv4 ACL is written as an `acl_spec`, which is a comma or tab-delimited string consisting of one or more `ace_specs`.
- A single NFSv4 ACE is written as an `ace_spec`, which is a colon-delimited, 4-field string in the following format:

    `type:flags:principal:permissions`

    Type is one of `A` (Allow), `D` (Deny), `U` (Audit), or `L` (Alarm).

    Flags may fall into four different categories:

        Used to indicate that the principle is a group.

        Used to indicate inheritance configuration for the ACL.

        Used to configure auditing of object access results.

        Flags may be empty.

    Principle is the user or group that the ACL applies to:

        Principle may also use three special tokens corresponding the typical POSIX owner, group owner, and world:

            OWNER@

            GROUP@

            EVERYONE@

    Permissions are a string of tokens that define access being denied or allowed to the principle:

        Notable permission tokens:

            `r`: read-data (files) / list-directory (directories)

            `w`: write-data (files) / create-file (directories)

            `a`: append-data (files) / create-subdirectory (directories)

            `x`: execute (files) / change-directory (directories)

            `d`:delete - delete the file/directory.

Examples:

Add ACE granting `alice@nfsdomain.org` generic *read* and *execute* access (defaults to prepending ACE to ACL) to the file foo:

```
nfs4_setfacl -a A::alice@nfsdomain.org:rxtncy foo
```

Edit existing ACL in a text editor and set modified ACL on clean save/exit:

```
nfs4_setfacl -e foo
```

- See `man nfs4_acl`, `man nfs4_setfacl`, and `man nfs4_getfacl` for more information.

CIFS Configuration

- The Concept
  - CIFS (part of the suite of Samba software) is designed to support Windows System mounting shares on a Linux Server.
  - CIFS maps windows credentials to Linux credentials.
  - Samba also comes with a daemon called windbindd that is used for integrating domain authentication into Linux.

    The service provided by winbindd is called `winbind` and can be used to resolve user and group information from a Samba or Windows NT server.

    The service can also provide authentication services via an associated PAM module, `pam_winbind`.

    Winbindd may also be integrated via `nsswitch.conf` through the service `winbind`(See objective 326.3 for more detail on `nsswitch.conf`).
  - The *smbd* and *nmbd* are the daemons that present CIFS shares.
  - The file `/etc/samaba/smb.conf` contains the primary configuration for samba.
  - More details may be found in `man samba`.

- Understand and use CIFS Unix Extensions
    - Note of Unix Extensions is made in `man smb.conf`
    - The Unix Extensions add enhanced features for Unix CIFS support such as symbolic links.
- Understand and configure CIFS security modes (NTLM, Kerberos).
    - CIFS may use Kerberos authentication or NTLM (windows style).
    - The `sec=` mount option may be provided to specify an authentication type.

      Some possible security options are:

        `krb5` for Kerberos authentication.

        `ntlm` for windows domain authentication.

        `none` no authentication (not recommended).
    - Bare in mind that for Kerberos authentication, your host must be properly configured in a Kerberos realm.
- Understand and manage mapping and handling of CIFS ACLs and SIDs in a Linux system
    - CIFS ACLs:

      The `cifsacl` mount option must be supplied when mounting for SID to UID mapping.

        `cifs.idmap` is used to map SIDs to UIDs (see `man cifs.idmap` for more detail).

        Winbindd must be configured in `nsswitch.conf`.

      `getcifsacl`, `setcifsacl` are used to control CIFS ACLs.

      `cifsacl` commands are provided in the `cifs-utils` package.

      CIFS ACLs support advanced windows permissions.

      How Access Control Entries for CIFS are defined:

        Every ACE entry starts with "ACL:" One or more ACEs are specified within double quotes.

        Multiple ACEs are separated by a comma.

The following fields of an ACE can be modified with possible values:

SID: Either a name or a raw SID value.

type: `ALLOWED (0x0)`, `DENIED (0x1)`, `OBJECT_ALLOWED (0x5)`, `OBJECT_DENIED (0x6)`

flags: `OBJECT_INHERIT_FLAG` (OI or 0x1), `CONTAINER_INHERIT_FLAG` (CI or 0x2), `NO_PROPAGATE_INHERIT_FLAG` (NI or 0x4), `INHERIT_ONLY_FLAG` (IO or 0x8), `INHERITED_ACE_FLAG` (IA or 0x10) or a combination/OR of these values.

mask: Either one of `FULL`, `CHANGE`, `READ`, a combination of `R W X D P O`, or a hex value

See `man setcifsacl` and `man getcifsacl` for more information.

- Relevant parameters for `mount.cifs`:

  `uid=arg`

  Sets the UID that will own all files or directories on the mounted filesystem when the server does not provide ownership information.

  When not specified, the default is UID 0.

  `forceuid`

  Instructs the client to ignore any UID provided by the server for files and directories and to always assign the owner to be the value of the `uid= option`.

  `cruid=arg`

  Sets the UID of the owner of the credentials cache.

  This is primarily useful with `sec=krb5`.

  The default is the real UID of the process performing the mount.

  Setting this parameter directs the `upcall` to look for a credentials cache owned by that user.

  `gid=arg`

  Sets the GID that will own all files or directories on the mounted filesystem when the server does not provide ownership information.

It may be specified as either a group name or a numeric GID.

When not specified, the default is GID 0.

The `mount.cifs` helper must be at version 1.10 or higher to support specifying the GID in non-numeric form.

`forcegid`

Instructs the client to ignore any GID provided by the server for files and directories and to always assign the owner to be the value of the `gid= option`.

`perm`

Client does permission checks (`vfs_permission` check of UID and GID of the file against the mode and desired operation).

Note that this is in addition to the normal ACL check on the target machine done by the server software.

Client permission checking is enabled by default.

`noperm`

Client does not do permission checks.

This can expose files on this mount to access by other users on the local client system.

Note that this does not affect the normal ACL check on the target machine done by the server software (of the server ACL against the username provided at mount time).

`dynperm`

Instructs the server to maintain ownership and permissions in memory that can´t be stored on the server.

This information can disappear at any time (whenever the inode is flushed from the cache), so while this may help make some applications work, its behavior is somewhat unreliable.

See `man mount.cifs` for more information.

Topic 328 Network Security

328.1 Network Hardening

Configure FreeRADIUS

- The Concept:
  - FreeRADIUS is a free and open-source policy server implementation for RADIUS.
  - RADIUS stands for Remote Authentication Dial-In User Service.
  - FreeRADIUS is used to authenticate network nodes typically to some backend authentication service.
  - It is popularly used for authentication to enterprise-class 802.1x networks and other such network authentication.
  - FreeRADIUS provides the `radiusd` daemon which provides an authentication service.
- Configuring FreeRADIUS:
  - The `freeradius` package provides the software required for FreeRADIUS.
  - Configuration files for radiusd exists in `/etc/raddb/*`
  - The primary configuration file is `/etc/raddb/radiusd.conf`

    Server logging and performance settings may be fine-tuned in this file.

    It is well commented.
  - Each client must be configured on the server using `/etc/raddb/clients.conf`

    Important configurations to be aware of:

      `ipaddr` — Specifies the client IP.

`secret` — The shared secret used to encrypt and decrypt communication between client and server.

Example configuration:

```
client private-network-1 {
  ipaddr          = 192.0.2.0/24
  secret          = testing123-1
}
```

- Administering FreeRADIUS:
  - `radmin` is used to connect to, and administer a running RADIUS server.

    See `man radmin` for a complete list of administration commands.
  - Additional utilities ship with the `freeradius-utils` package.
  - The configuration may be tested using the `radtest` command.

    It is a wrapper around `radclient` which may be used to interface with a RADIUS server.

    Using `radtest`

    `radtest user password radius-server nas-port-number secret`

    `radclient` is the client-side program used to interact with a RADIUS server.
  - `radlast` is a wrapper around the `last` command that targets `/var/log/radius/radwtemp` instead of the default.
  - `radwho` shows currently logged on users.
  - Man pages are available for `radtest`, `radclient`, `radlast`, and `radwho` and provide more detailed information.

Capturing and analyzing network traffic

- A common utility used for capturing network traffic is `tcpdump`:
  - The `tcpdump` command will write all incoming and outgoing packet information to stdout by default.
  - The `-w` flag may be used to specify and output files for the data.
  - The `-i` flag may be used to have `tcpdump` record packet info from only certain interfaces.
  - The `tcpdump` utility may also take filters to limit what packet information is logged.

    Expressions are provided optionally and require no flag.

    The expressions take the same form as the capture filters used by wireshark (see below section).
  - See `man tcpdump` for more detail.
- Another utility that is useful for network troubleshooting is `Wireshark`.
- Wireshark is a GUI application that is more robust than tcpdump and includes the following features:
  - Captures network traffic.
  - Analyzes network traffic using display filters and a number of built-in reports:

    See statistics menu in the program.
  - Capture Filters in Wireshark:

    Capture only traffic to or from IP address 172.18.5.4:

    `host 172.18.5.4`

    Capture traffic to or from a range of IP addresses:

    `net 192.168.0.0/24`

    `net 192.168.0.0 mask 255.255.255.0`

    Capture traffic from a range of IP addresses:

    `src net 192.168.0.0/24`

`src net 192.168.0.0 mask 255.255.255.0`

Capture traffic to a range of IP addresses:

`dst net 192.168.0.0/24`

`dst net 192.168.0.0 mask 255.255.255.0`

Capture only DNS (port 53) traffic:

`port 53`

Capture a range of TCP ports:

`tcp portrange 1501-1549`

`portrange 1501-1549 and tcp`

- While wireshark is a GUI tool, it has a CLI equivalent knowns as **tshark**.
  - Without any switches, `tshark` performs similarly to `tcpdump`
  - Key `tshark` switches.

    `-r file`: Read in `pcap` file.

    `-i eth0`: Capture specific interface.

    `-w file`: Output `pcap` to a file.

    `-T <type>`: Set output type (fields, json, etc)

    `-e <field>`: Provide a field to display

    `-z <report>`: Provide various available reports.
  - See `man tshark` for more detail.

Network Utilities

- Working with the `nmap` command
  - `nmap` stands for Network Mapper.
  - It is used for network exploration or security auditing.
  - `nmap` uses raw IP packets in novel ways to determine the following:

    What hosts are available on the network.

    What services (application name and version) those hosts are offering.

    What operating systems (and OS versions) they are running.

    What type of packet filters/firewalls are in use.
  - Scan Networks and hosts:

    `nmap -sn 10.0.0.0/24`

    Pings all host in 10.0.0.0/24 and list hosts who respond.
  - Different scan methods listed by `nmap switch: <type of scan>`:

    `-sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans`

    `-sU: UDP Scan`

    `-sN/sF/sX: TCP Null`,`FIN`, and `Xmas scans`

    `--scanflags <flags>: Customize TCP scan flags`

    `-sI <zombie host[:probeport]>: Idle scan`

    `-sY/sZ: SCTP INIT/COOKIE-ECHO scans`

    `-sO: IP protocol scan`

    `-b <FTP relay host>: FTP bounce scan`

- Notable options
  - `-T#`
    - The # may be 1 through 5.
    - This dictates how fast `nmap` will work.
    - The higher the number, the quicker the scan.
    - Running faster scans may produce less results.
  - `-A`: Enable OS detection, version detection, script scanning, and traceroute
  - `-P`: Control ping types used.
  - `-s`: Dictate scan options (see scan method section above).
- Examples:
  - `nmap -A -T4 scanme.nmap.org`
  - Perform OS detection, version detection, script scanning, and traceroute (`-A`) with T4 timing against `scanme.nmap.org`.
  - `nmap -sS -T3 scanme.nmap.org`
    - Perform SYN scan with T3 timing against `scanme.nmap.org`.

Neighbor Discovery Protocol Monitor (NDPMon) is a diagnostic software application used by Internet Protocol version 6 network administrators for monitoring ICMPv6 packets.

- NDPMon observes the local network for anomalies in the function of nodes using Neighbor Discovery Protocol (NDP) messages, especially during the Stateless Address Autoconfiguration.
- When an NDP message is flagged, it notifies the administrator by writing to the syslog or by sending an email report.

Identify and deal with rogue router advertisements and DHCP messages:

- Rogue router advertisements is a problem that is related to IPv6.

There are relevant parameters that may be tuned in `/proc/sys/net/ipv6`:

- In particular, the following should be set to 0 for all interfaces:
  - `/proc/sys/net/ipv6/conf/<interface>/forwarding`
  - `/proc/sys/net/ipv6/conf/<interface>/accept_ra`

Rouge DHCP messages are another network issue:

- They are mitigated by restricting DHCP messages at the switch level.
- This is done using DHCP snooping which restricts which ports may originate DHCP broadcast messages.

328.2 Network Intrusion Detection

Network Monitoring

- `ntop` is a network traffic probe that provides network usage information.
  - The `ntop` command may be used for basic administration tasks or to launch the ntop daemon:
    - Notable options of the `ntop` command:
      - `--set-admin-password=password`
      - Set the administrator password for `ntop` to *password*
    - Running the ntop daemon:
      - Start ntop: `ntop -P /etc/ntop -W4242 -d`
        - `-P` option reads the configuration files in the “/etc/ntop” directory.
        - The `-W` option enables the port on which we want to access ntop through our web browser:
          - If you don't specify this option, the default port is 3000.
      - The `-d` option enables ntop in daemon mode.

Once is started in web mode, ntop enables its web server and allow us to view and use its statistics through any web browser.

Access web console by http://host:portnumber/.

- Cacti
    - Provides network monitoring with a web-based interface.
    - Provides graphing functionality.
    - More of a general purpose monitoring tool compared to ntop.

Configure and use Snort, Including Rule Management

- The Concept:
    - Snort is a software package that has three primary functions:

      A packet sniffer like tcpdump

      A packet logger (useful for network traffic debugging, etc.)

      A full feature network intrusion prevention system
    - Snort is available in a commercial version as well as a community capacity for no cost.
- Install snort:
    - Check out https://www.snort.org/#get-started for latest packages and distributions.
    - Packages not generally available in typical repositories.
    - Requires 2 packages (available from *snort.org*):

      Snort base packages - Package: `snort`.

      Data acquirer (daq) - Package: `daq`
    - Snort configuration and rules exists in `/etc/snort/*`.
    - The file `/etc/snort/snort.conf` allows for the definition of variables and general snort configuration.

- The `snort` command:

    Ran without options, the `snort` command captures network traffic and provides a brief summary when it exits.

    The daemon mode is initiated with the `-D` switch.

- The `snort-stat` command

    Not present in the latest snort releases.

    Reads syslog files containing Snort information and generates port scan statistics.

- Snort Rules

    Snort Rules are made up of two primary components: The Rule Header and The Rule Option.

    Snort Rule Template: `[action][protocol][sourceIP][sourceport] -> [destIP][destport] ( [Rule options] )`

        The header is the part prior to the parenthesis.

        The Options are enclosed in the parenthesis.

    Rule Header Format:

        `action`

            Action to take (option)

            The rule action tells Snort what to do when it finds a packet that matches the rule criteria (usually alert).

            Actions:

                `alert` — Generate an alert based on configuration and log packet

                `pass` — Do not intercept

                `log` — Log packet

                `activate` — Alert and turn on dynamic rule

                `dynamic` — A rule that may only be enabled by `activate` `reject` — Drop packet, log packet, and send TCP reset on TCP connections and ICPM port unreachable for UDP connections

`drop` — Drop packet

`sdrop` — Drop packet and do not log

protocol

Type of traffic.

There are four protocols that Snort currently analyzes for suspicious behavior: TCP, UDP, ICMP, and IP.

Source address(es)

Traffic source address(es).

May be variable or literal.

Source port(s)

Traffic source port(s).

May be variable or literal.

->

Direction operator

The direction operator -> indicates the orientation of the traffic to which the rule applies.

Destination address(es)

Traffic destination address(es).

May be variable or literal.

Destination port(s)

Traffic destination port(s).

May be variable or literal.

- Rule Options
  - Snort rule options are separated from each other using a semicolon.
  - Rule option keywords are separated from their arguments with a colon.
  - There are two types of rule option: General and Detection.
  - General Rule Options
    - Message
      - A meaningful message typically includes what the rule is detecting.
      - It is a simple text string.
    - Flow
      - For the rule to fire, specifies which direction the network traffic is going.
      - Used in conjunction with TCP stream reassembly.
      - It allows rules to only apply to certain directions of the traffic flow.
    - Reference
      - Allows rules to include references to external sources of information (such as a URL).
    - Classtype
      - How Snort shares what the effect of a successful attack would be.
    - sid/rev
      - The snort id is a unique identifier for each rule.
      - This information allows output plugins to identify rules easily and should be used with the rev (revision) keyword.
  - Detection Rule Options
    - Content
      - Allows the user to set rules that search for specific content in the packet payload and trigger response based on that data.

May contain mixed text and binary data.

distance/offset

Allow the rule writer to specify where to start searching relative to the beginning of the payload or the beginning of a content match.

within/depth

Allow the write rule to specify how far forward to search relative to the end of a previous content match and, once that content match is found, how far to search for it.

PCRE

Allows rules to be written using Perl compatible regular expressions which allows for more complex matches than simple content matches.

Byte test.

Allows a rule to test a number of bytes against a specific value in binary.

Configure and use OpenVAS, including NASL

- The Concept:
  - OpenVAS provides network vulnerability scanning and vulnerability management.
  - Crucially important to staying ahead of vulnerabilities is keeping NVT (network vulnerability tests).
  - Nessus Attack Scripting Language (NASL) is generally used when creating vulnerability tests.

    NASL is a proprietary scripting language.

    Typically uses include:

      NVT development

      Custom testing scenarios

- Install and Configure OpenVAS Scanner:
  - Install `openvas-scanner` for openvassd
  - The `openvassd` program is the daemon for the openvas scanner.
  - Configuration files located at `/etc/openvas/openvassd.conf`.
  - Must run `openvas-mkcert -q` to establish scanner certificate.

    See 'securing scanner communication' below.
  - `systemctl enable openvas-scanner && systemctl start openvas-scanner`
- Administering OpenVAS
  - Keeping up to date on threats:

    The OpenVAS Security Scanner performs several security checks, each of them being coded as an external plugin coded in NASL.

    As new security holes are published every day, new plugins appear on the OpenVAS site (*www.openvas.org*).

    The script `openvas-nvt-sync` will fetch all the newest security checks for you and install them at the proper location.

    Requires restarting openvas-scanner(8) or send a SIGHUP to its main process.

    `openvas-nvt-sync` uses rsync(1) and md5sum(1) to do its job.

    Host should have access to rsync.openvas.org using the rsync protocol (TCP/UDP port 873).
  - Securing Scanner Communication:

    The OpenVAS Scanner protects its communication with clients by using SSL.

    SSL requires the scanner to present a certificate to the client.

    The client can optionally present a certificate to the scanner.

    `openvas-mkcert` creates a certificate authority (if none:q exists already) and generates the scanner certificate.

      The `-q` option will not prompt for input and select defaults.

328.3 Packet Filtering

Firewall Review

- Firewall essentials are covered in LPIC-2 202 but are reinforced in LPIC-3 303.
- LPIC-3 303 addresses some more advanced topics such as:
  - IP Sets
  - DMZ Zones
  - Connection Tracking
  - Network Address Translation
  - Bridged Network Filtering with ebtables
  - Alternatives to iptables such as nftables
- The Relationship between netfilter, iptables, nftables (basic), and ebtables (basic).
- Essential packet filtering (LPIC-2 202 Review)
  - Netfiler is a kernel module used to manage the kernel firewall.
  - The `iptables` command operates through netfilter on a series of tables that each have chains affiliated with them.
  - Each chain consists of a set of rules that determine the action to take on a given packet based on provided criteria.
  - The tables are *filter*, *nat*, *mangle*, and *raw*.
  - Chains include options such as *INPUT*, *OUTPUT*, *PREROUTING*, and *POSTROUTING*.
  - Command example: `iptables -t filter -A INPUT -i eth0 --dport 22 -j ACCEPT`

    `-t filter -A INPUT` means it will use the INPUT chain on the filter table.

    `-i eth0` matches any traffic coming through eth0

`--dport 22`matches any traffic with a destination port of 22

`-j ACCEPT` indicates to accept traffic matching the provided criteria.

- The `iptables-save` and `iptables-restore` commands can save running rule sets to files and restore rule sets from files.
- The `ip6talbles` and related commands operating on the ipv6 stack just like their ipv4 counterparts.
- Review LPIC-2 202 section 212 for more detailed information.

Advanced Firewall Concepts

- IP Sets
  - `ipset` is used to set up, maintain, and inspect so called IP sets in the Linux kernel.
  - Depending on the type of the set, an IP set may store (among other things):

    IP(v4/v6) addresses

    (TCP/UDP) port numbers

    IP and MAC address pairs

    IP address and port number pairs
  - `Iptables` matches and targets referring to sets create references, which protect the given sets in the kernel.
  - A set cannot be destroyed while there is a single reference pointing to it.
  - See `man ipset` for more information.
- Firewall DMZ
  - A firewall DMZ (short for demilitarized zone) provides additional security for a network.
  - A DMZ is a subnet that is separate from a general LAN.
  - The DMZ may be accessed by external networks and is where services and hosts that require a public face are located on the network.

    - The LAN is completely firewalled off from external network access for enhanced security.
- Connection Tracking
    - Often it is necessary for a firewall to track a connection state.
    - `conntrackd` is a daemon that works with netfilter to manage connection states.
    - See `man conntrackd` for more information.
- Network Address Translation
    - For load balanced services or in cases where many services are hosted behind a single public IP address, it is necessary to translate IP addresses.
    - Firewalls and routers can handle this translation.
    - This is managed using the nat table in iptables and the PREROUTING and POSTROUTING chains.
- Bridged Network Filtering with ebtables
    - Ethernet frames may be inspected and filtered used the `ebtables` command.
    - The `ebtables` command is analogous to `iptables`, however, it only operates on tables specific to ethernet bridges:

        `filter`

        `broute`

        `nat`
    - Notable `ebtables` options

        `-Ln`: Lists line numbers when printing a rule set.

        `-Lc`: Lists packet and byte counters with each rule.

Nftables

- Alternatives to iptables (that matter on LPIC-3 303)
  - One of the notable drawbacks of iptables is the complexity of it.
  - Several alternatives have been developed but LPIC-3 303 is only concerned with nftables.
  - Nftables aims to provide a simpler interface to netfilter.
  - The `nft` command is how we can interact with nftables.
  - Nftables uses a different syntax from iptables in an effort to simplify the rules sets.
  - Nftables is compatible with iptables style rules.
  - Only brief familiarity required for the exam.

328.4 Virtual Private Networks

OpenVPN server

- Configuring OpenVPN server
  - Software provided by `openvpn` package.
  - Endpoints are secured by `ssl certs` (like Apache).
  - Primary configuration file is `/etc/openvpn/server.conf`:

    Example may be found at:
    `/usr/share/doc/openvpn-<version>/sample/sample-config-files/server.conf`
  - Important configurations:

    Must open port on which the server will operate.

    Provide certificates (ca cert and key directives).

May use relative paths, the software looks in `/etc/openvpn` for files provided in this way.

- Enabling and starting OpenVPN:

  `systemctl enable openvpn@server`

  `systemctl start openvpn@server`

- Firewall rules for `openvpn`

  Do not drop VPN network traffic.

  `iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j MASQUERADE`

- Enable ipv4 forwarding:

  `sysctl -w net.ipv4.ip_forward=1`

- Configuring OpenVPN Client
  - Configurations exists in `/etc/openvpn/client.conf`
  - Example may be found at: `/usr/share/doc/openvpn-<version>/sample/sample-config-files/client.conf`
  - Enabling and starting OpenVPN:

    `systemctl enable openvpn@client`

    `systemctl start openvpn@client`

- `openvpn` options
  - All timeout configurations are in seconds.
  - `--mlock` prevents writing of sensitive key information to virtual memory (and subsequently hard disk) which enhances security.
  - `--push` will push configuration files to clients.
  - Be familiar with the general format of `openvpn` options.
  - See `man openvpn` for more information.

Working with IPsec server and clients

- Using IPSec for VPN networks
  - The Concept:

    IPSec is used to create a peer to peer secure connection.

    The Kernel maintains two databases for use IPsec:

      The *Security Association Database(SAD)*:

        A *Security Association (SA)* describes how entities will use security services to communicate.

        SAD entries contain the key of each IPsec-SA.

      The Security Policy Database(SPD):

        Kernel refers to SPD in order to decide whether to apply IPsec to a packet or not.

        SPD entries specify which/how IPsec-SA is applied.

    The `ipsec-tools` package includes utilities needed for establishing such a connection.

      `setkey` adds, updates, dumps, or flushes Security Association Database (SAD) entries as well as Security Policy Database (SPD) entries in the kernel.

      Notable `setkey` syntax:

        Use `-f` to supply a file containing database entries.

        Use `-c` to supply database entries via stdin.

        `add`, `get`, `delete`, `flush`, `dump` all manipulate the SAD database entries.

        `spdadd`, `spdupdate`, `spddelete`, `spdflush`, `spddump` all manipulate the SPD database entries

      Racoon is an IKEv1 keying daemon.

    L2TP is a VPN technology:

      It is insecure by itself.

It must be used over an IPSec link due to this.

Configurations may be found in:

`/etc/ipsec-tools.conf`

`/etc/racoon/racoon.conf`