Red Hat Identity Management (IdM) - FreeIPA Identity Policy Audit

IdM server with integrated DNS using FreeIPA which incorporates Kerberos, LDAP, TLS CA, NTP, and BIND in one install.

Installation:

Begin with getting the needed packages:

sudo dnf install freeipa freeipa-server bind bind-utils bind-dyndb-ldap krb5-server krb5-libs chrony

Next, run the installer script. This provides integrated DNS which will be relied on by other FreeIPA components

sudo ipa-server-install --enable-dns (--enable-dns is not needed in RHEL8 and beyond)

During the run of the installer script, you will be asked for the domain name (realm), you'll need to set a Directory Manager password and a primary administrator password. You will also be asked for DNS settings like type (usually choose BIND, if it asks to), forwarders (8.8.8.8 is fine for non-enterprise or testing installs), and reverse DNS (use the in-addr.arpa reverse version of our primary IP address, i.e., 192.168.1.123 would be 123.1.168.192.in-addr.arpa). You can also expect to see some choices and setting for Kerberos integration, NTP , database type (often PostgreSQL), and LDAP.

Here is what we have after that completes:
Dogtag Certificate System: Certificate Authority & Registration Authority for certificate management
LDAP Server: Employs 389 Directory Server for user and group management.
MIT KDC: Kerberos Key Distribution Center is the basis for single sign-on
Apache: IdM administration functionalities need a built-in webserver, so there it is.
NTP (chrony in RHEL 8/9): Sets up the Network Time Protocol service
BIND: Integrates the BIND DNS server with the FreeIPA environment for DNS management
SSSD - client side component employing FreeIPA as authentication & identity provider superior to NSS & PAM.

FreeIPA SystemD Services:
FreeIPA Server: freeipa-server.service, ipactl.service, freeipa-healthcheck.service
Kerberos Key Distribution Center: krb5kdc.service
Kerberos DB Administration: kadmin.service
Directory Services (LDAP): slapd.service
DNS Server: named.service

Important Configuration Files:
The primary config file - /etc/ipa/default.conf

Server Settings:

realm (Required): Defines your FreeIPA realm name (e.g., EXAMPLE.COM).
server_principal: (Optional) Specifies name for FreeIPA server. Autogenerates as host/<hostname>@<realm>
server_cert: (Optional) Path to FreeIPA server certificate file
server_key: (Optional) Path to private key file associated with the server certificate
ca_cert: (Optional) Path to CA certificate used to sign the FreeIPA server certificate
offline: (Optional) True means disable communication with other FreeIPA servers (isolated deployments)

DNS Settings:

enable_dns: (Optional) Set to true to enable the integrated FreeIPA DNS server. Defaults to false
dns_forwarder: (Optional) Comma-separated list of IPs of DNS servers to forward unresolved queries to
dns_allow_update: (Optional) IPs or networks allowed to update DNS records. Default is 127.0.0.1
disable_anonymous_bind: Restrict anonymous BIND queries, improving security
forwarder_permit: Define IPs or networks allowed for DNS forwarding requests (prevents open relays)

Security Settings:

password_minimum_length: Set a minimum password length
password_require_mixed_case: Require mixed case (uppercase/ lowercase)
password_require_numeric: Require at least one number
password_require_special: Require at least one special character
user_enable_lockout: Enable user account lockout after failed login attempts
user_lockout_duration: Define the duration (minutes) a locked account remains inaccessible
allow_unsafe_kerberos_keytypes: Leaving this disabled prevents weak Kerberos encryption types
ca_cert_subject: Defines the subject info for a custom CA certificate
server_cert_subject: Defines the subject info for a custom FreeIPA server certificate
db_type: Specifies the database backend used by FreeIPA (defaults to postgresql)
allow_weak_password: Disabling this enforces strong passwords for IPA clients (keep false).

Debug and Logging:

debug_level: Sets the debug logging level (higher values provide more detailed logs).
log_file: Path to the log file for FreeIPA server events.

[For very specialized configs, an optional /etc/ipa/server.conf can be used for server-specific overrides. it would be read first but is seldom needed and this is simply a footnote to it "being a thing"]

Enforcing standardized user authentication with /etc/ipa/userauth.conf

A general idea of entiries in a /etc/ipa/userauth.conf file. Security management could mandate this be used to emphasize and/or standardize password policy and security configurations (sometimes simple alternates to kerberos). The details for each module is a little out of scope for this writing, but module docs would have specifics and actual items to replace what's below. Many configuration options will be in a module's config file.

Options in /etc/ipa/default.conf can also be over-ridden here, and there are some new ones:
password_history_depth: Define the number of previous passwords a user cannot reuse.
user_enable_lockout: Enable account lockout after a certain number of failed login attempts.
user_lockout_duration: Define the duration (minutes) an account remains locked after failed login attempts.

```
[Service: sudo]      #  Enable RADIUS authentication for sudo service
    authtype = radius
    server = radius.example.com  # Replace with actual RADIUS server address
    shared_secret = (secret)     # Replace with actual shared secret (not recommended in plain text)
    port = 1812                  # Default RADIUS port
    #   timeout = 3                # RADIUS authentication timeout (seconds)
    #   nas_port_type = 5          # Network Access Server (NAS) port type

[Service: vpn]      #  Enable LDAP authentication for a custom VPN service
    authtype = ldap
    server = ldap.example.com    # Replace with actual LDAP server address
    basedn = dc=example,dc=com   # Replace with appropriate base DN for user search
    binddn = cn=FreeIPA_Bind_User,ou=Service Accounts,dc=example,dc=com  # Replace with bind DN
    bind_password = (secret)     # Replace with actual bind password (not recommended in plain text)
    #   search_scope = subtree       # LDAP search scope (base, onelevel, subtree)
    #   tls_cacertfile = /etc/ipa/certs/ca.crt  # Path to CA certificate for LDAP TLS

[Service: shell]      #  PAM for shell logins
    auth         pam_ServiceName.so

[Service: secureapp]      #  Enable token-based authentication for a specific application (hypothetical)
    authtype = token  # Assuming a token-based authentication module is installed

[Service: shell]      #  Disable alternative authentication for shell logins (only use Kerberos)
    alternative_authentication = false

[Service: console]      #  PIN login module.  Allows users to log in using a PIN instead of a password
    authtype = pin  # Assuming the PIN login module is installed
    #   pin_retries = 3  # Maximum allowed PIN attempts before lockout
    #   pin_length = 6   # Minimum PIN length

[Service: ssh]      #  2FA/OTP (Example: Google Authenticator- others include RSA SecurID, Duo Security, etc.
    require_otp = true  # Enforces OTP for SSH logins
    #  require_mfa = true  # Enforces MFA for SSH logins

[Service: myapp]      #  Social login module (hypothetical - Facebook for a custom web application):
    authtype = social  # Assuming a social login module is installed
    #   provider = facebook  # Specify Facebook as the social login provider
    #   client_id = your_facebook_app_client_id  # Replace with your Facebook App details
    #   client_secret = (secret)  # Replace with your Facebook App secret (avoid plain text)

[Service: sudo]      #  Certificate-based auth module using PKI for sudo service (Example: freeipa-certlogin):
    authtype = cert  # Assuming the freeipa-certlogin module is installed
    #   ca_certfile = /etc/ipa/certs/ca.crt  # Path to the Certificate Authority certificate
    #   require_crl_check = true  # Enforce Certificate Revocation List (CRL) checking

[Service: shell]      #  External database auth module (example: ipa_ldap_sync-  LDAP for shell logins):
    # Users are authenticated against FreeIPA, but user data is synchronized from LDAP server
    uri = ldaps://ldap.example.com:636
```

The IPA commands for user and resource management
ipa <category> <subcommand> [options] [arguments]
<category> is for example user, group, host, etc.) <subcommand> is the action (e.g., add, delete, show, etc.)
Most of categories typically have the subcommands add, delete, modify, show, show all (or list), and find

ipa config: Manage FreeIPA server configuration files
ipa package: Manage FreeIPA packages (installation, updates)
ipa profile: Manage FreeIPA server profiles (configurations)
ipa server: Manage the FreeIPA server itself (installation, configuration)
ipa vpnconfig: Manage VPN configuration options within FreeIPA
ipa trust: Manage trust relationships (e.g., with Active Directory)
ipa host: Manage FreeIPA hosts (machines joining the identity domain)
ipa hostgroup: Manage groups specifically for FreeIPA hosts (machines)
ipa interface: Manage network interfaces on the FreeIPA server
ipa nfsserver: Manage FreeIPA's NFS server configuration
ipa service: Manage FreeIPA services (applications requiring identity management)
ipa join: Joins a machine to a FreeIPA domain without using the client installation command.
ipa domain: Manage FreeIPA domains (logical groupings of identities)
ipa fqdn: Manage Fully Qualified Domain Names (FQDNs) associated with FreeIPA
ipa domaindns: Manage DNS domains integrated with FreeIPA
ipa dnskey: Manage DNS keys used for DNS signing (important for DNSSEC)
ipa dbbackup: Manage database backups and restores
ipa dnstable: Manage FreeIPA's internal data tables (use with caution)
ipa restore/backup: Create or load a backup of an IPA config into FreeIPA
ipa sync: Synchronize data with external directory services
ipa topology: Manage FreeIPA's server topology (replica management)
ipa vault: Manage FreeIPA vaults (secure storage for secrets)
ipa ca: Manage Certificate Authority operations (for internal PKI)
ipa cert: Manage certificates used by FreeIPA (server TLS, user certificates)
ipa tls: Manage Transport Layer Security (TLS) certificates
ipa kerberos: Manage Kerberos tickets and keytabs
ipa servicedelegationrule: Manage service delegation rules (allow services to request Kerberos tickets)
ipa servicedelegationtarget: Manage service delegation targets (used with service delegation rules)
ipa realmdomains: Manage realm domains used for Kerberos authentication
ipa diagnose: Perform diagnostic operations on the FreeIPA server
ipa monitor: Monitor the FreeIPA server's health and status
ipa find: Search for users, groups, hosts, and other FreeIPA objects
ipa user: Manage FreeIPA users - ipa userpolicy: Manage user password policies - ipa group: Manage groups
ipa passwd: Reset or change passwords for FreeIPA users and services
ipa pwpolicy (alias for userpolicy): Manage password policies (password complexity)
ipa shadow: Manage shadow password information (use with caution)
ipa permission: Manage individual permissions assigned to users or groups
ipa rightsource: Manage rights sources used for access control
ipa role: Manage FreeIPA roles (sets of permissions)
ipa relation: Manage relationships between FreeIPA objects (e.g., user-group membership)
ipa selinuxusermap: Manage SELinux user maps
ipa sshkey: Manage SSH keys for FreeIPA users and services
ipa hbacrule: For Host-Based Access Control (HBAC) - ipa hbacsvc (services) - ipa hbacsvcgroup (groups)
ipa sudocmd: Manage commands usable w/ sudo - ipa sudocmdgroup: for sudo command groups
ipa idrange: Manage ID ranges (used for ID mapping)
ipa locale: Manage locales used within the FreeIPA server

For details, you can use "ipa help <category>" for any of these.

Open ports for FreeIPA functionality:
TCP ports: 80, 443 (HTTP/S for web interface), 389, 636 (LDAP/S), 53 (DNS), 88 (Kerberos for Windows clients)
UDP ports: 88 (Kerberos), 53 (DNS), 67 and 68 (DHCP)
RPC and rstatd use random port numbers. Unless you have multiple FreeIPA servers or modules that need it, you are probably safe to not worry about opening ports- addressing this issue is outside the scope of this writing.

Important files and directories
/etc/ipa: This directory contains configuration files for FreeIPA.
/etc/ipa/client.conf: optional- for FreeIPA client on the server itself. Has location of the server and realm info
/etc/ipa/userauth.conf (Optional) - Defines authentication backends and policies for user login.
/etc/ipa/authpolicy.conf (Optional) - Configures authentication policy for FreeIPA services.
/etc/ipa/db.conf - If you have database configuration info outside of default.conf they would go in this

/var/lib/ipa: This directory contains data files for FreeIPA, including LDAP databases and Kerberos keytabs.
/etc/krb5.conf: Config for the Kerberos client, with location of Kerberos keytabs and realm information.
/etc/pki/pki-tomcat: Directory for the Dogtag CA Certificate Authority service
/etc/pki/pki-tomcat/alias: Contains the certificate database used by Dogtag.
/etc/ipa/certs/: Holds FreeIPA server user and service certificates, private keys
/root/cacert.p12: Admin access certificate - default name, PKCS#12 for Public Key Cryptography Stds #12
/etc/ipa/ca.crt: The CA certificate file used by clients to verify the FreeIPA server's identity.
/etc/ipa/nssdb: Contains the NSS (Network Security Services) database used for storing certificates and keys.
/var/lib/freeipa/dns/: Holds zone files managed by FreeIPA's internal DNS server. Each domain/subdomain gets a zone file.
The TLD zone file is the primary, containing records for users, computers, and other services.
FreeIPA manages user/group information and dictates DNS records while BIND takes directions

/var/log/ipa: This directory contains log files related to FreeIPA operations.
/var/log/ipa-server-install.log: Log file for FreeIPA server installation.
/var/log/ipa-client-install.log: Log file for FreeIPA client installation.

Applying basic system security mechanisms:

Hardening FreeIPA with SELinux
Install the package: dnf install policycoreutils-selinux-freeipa
SELinux types and contexts:
- ipa_var_lib_t: Files under /var/lib/ipa
- ipa_var_run_t: Files under /var/run/ipa
- ipa_log_t: Logs under /var/log/ipa
- ipa_tmp_t: Temporary files
- ipa_exec_t: Executable files

Applying contexts and set booleans:

Item	Context to apply	Save what was applied
/var/lib/ipa	semanage fcontext -a -t ipa_var_lib_t "/var/lib/ipa(/.*)?"	restorecon -Rv /var/lib/ipa
/var/run/ipa	semanage fcontext -a -t ipa_var_run_t "/var/run/ipa(/.*)?"	restorecon -Rv /var/run/ipa
/var/log/ipa	semanage fcontext -a -t ipa_log_t "/var/log/ipa(/.*)?"	restorecon -Rv /var/log/ipa
/tmp/ipa	semanage fcontext -a -t ipa_tmp_t "/tmp/ipa(/.*)?"	restorecon -Rv /tmp/ipa
Executables	semanage fcontext -a -t ipa_exec_t "/usr/libexec/ipa(/.*)?"	restorecon -Rv /usr/libexec/ipa
Allow LDAP over SSL (boolean)	setsebool -P allow_ipa_ldap_ssl 1	

Verifying SELinux contexts and booleans:
ls -Z /var/lib/ipa && getsebool -a | grep ipa

To identify and resolve denials:
grep "denied" /var/log/audit/audit.log | audit2allow -M mypol
semodule -i mypol.pp

Example configuration for firewalld:
```
firewall-cmd --zone=internal --add-source=10.0.10.0/24 --permanent
firewall-cmd --zone=internal --add-source=172.16.20.0/24 --permanent  # Add 2nd subnet to internal zone
firewall-cmd --zone=internal --add-port=443/tcp --permanent  # HTTPS
firewall-cmd --zone=internal --add-port=80/tcp --permanent    # Optional for web interface
firewall-cmd --zone=internal --add-port=389/tcp --permanent  # LDAP
firewall-cmd --zone=internal --add-port=636/tcp --permanent  # LDAPS
firewall-cmd --zone=internal --add-port=88/udp --permanent  # Kerberos
firewall-cmd --zone=internal --add-port=88/tcp --permanent  # Kerberos (optional for Windows clients)
firewall-cmd --zone=internal --add-port=53/udp --permanent  # DNS
firewall-cmd --zone=internal --add-port=53/tcp --permanent  # DNS
firewall-cmd --zone=internal --add-port=67/udp --permanent  # DHCP server broadcasts
firewall-cmd --zone=internal --add-port=68/udp --permanent  # DHCP clients leases
firewall-cmd --permanent --default-zone=internal    # Set internal zone as default
firewall-cmd --reload    # Reload firewall configuration
```

Example configuration for Iptables:

```
# Chain for internal subnet 1 traffic
iptables -A INPUT -i eth0 -s 10.0.10.0/24 -p tcp --dport 443 -j ACCEPT
iptables -A INPUT -i eth0 -s 10.0.10.0/24 -p tcp --dport 80 -j ACCEPT  # for web interface
iptables -A INPUT -i eth0 -s 10.0.10.0/24 -p tcp --dport 389 -j ACCEPT
iptables -A INPUT -i eth0 -s 10.0.10.0/24 -p tcp --dport 636 -j ACCEPT
iptables -A INPUT -i eth0 -p udp -s 10.0.10.0/24 --dport 88 -j ACCEPT  # Kerberos
iptables -A INPUT -i eth0 -p tcp -s 10.0.10.0/24 --dport 88 -j ACCEPT  # Kerberos (optional for Windows clients)
iptables -A INPUT -i eth0 -p udp -s 10.0.10.0/24 --dport 53 -j ACCEPT  # for DNS
iptables -A INPUT -i eth0 -p tcp -s 10.0.10.0/24 --dport 53 -j ACCEPT  # for DNS
iptables -A INPUT -i eth0 -p udp -s 10.0.10.0/24 --dport 67 -j ACCEPT   # see DHCP server broadcasts
iptables -A OUTPUT -i eth0 -p udp -s 10.0.10.0/24 --sport 67 --dport 68 -j ACCEPT  # DHCP leases, broadcasts
# Allow established connections for subnet 1
iptables -A INPUT -i eth0 -s 10.0.10.0/24 -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -o eth0 -d 10.0.10.0/24 -m state --state ESTABLISHED,RELATED -j ACCEPT

#  Chain for internal subnet 2 traffic
iptables -A INPUT -i eth0 -s 172.16.20.0/24 -p tcp --dport 443 -j ACCEPT
iptables -A INPUT -i eth0 -s 172.16.20.0/24 -p tcp --dport 80 -j ACCEPT  # for web interface
iptables -A INPUT -i eth0 -s 172.16.20.0/24 -p tcp --dport 389 -j ACCEPT
iptables -A INPUT -i eth0 -s 172.16.20.0/24 -p tcp --dport 636 -j ACCEPT
iptables -A INPUT -i eth0 -p udp -s 172.16.20.0/24 --dport 88 -j ACCEPT  # Kerberos
iptables -A INPUT -i eth0 -p tcp -s 172.16.20.0/24 --dport 88 -j ACCEPT  # Kerberos (for Windows clients)
iptables -A INPUT -i eth0 -p udp -s 172.16.20.0/24 --dport 53 -j ACCEPT  # for DNS
iptables -A INPUT -i eth0 -p tcp -s 172.16.20.0/24 --dport 53 -j ACCEPT  # for DNS
iptables -A INPUT -i eth0 -p udp -s 172.16.20.0/24 --dport 67 -j ACCEPT   # see DHCP server broadcasts
iptables -A OUTPUT -i eth0 -p udp -s 172.16.20.0/24 --sport 67 --dport 68 -j ACCEPT  # send DHCP leases, BC
# Allow established connections for subnet 2
iptables -A INPUT -i eth0 -s 172.16.20.0/24 -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -o eth0 -d 172.16.20.0/24 -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Post-installation checklist:

- Lock down the FreeIPA system with the options provided in the hardening section
- Create admin accounts with strong passwords for managing the FreeIPA domain. Organize them into groups
- Set up other users and groups to maintain consistency and organization within your domain.
- Define settings like lifetime for certificates issued by your CA if you're using an internal PKI for authentication.
- Issue server certificates for the FreeIPA server and others for secure communication within the domain.
- If needed, integrate FreeIPA with your existing DNS infrastructure for automatic DNS record management.
- Define Kerberos realm settings within FreeIPA if you plan to use Kerberos for authentication.
- Enroll machines (clients and servers) into the FreeIPA domain using the ipa join command.
- Install and configure FreeIPA client software on domain members
- Implement monitoring/logging solutions for server activity, identify potential issues, and ensure secure operation.
- Establish a regular backup and restore strategy for server configuration and data for disaster recovery
- Create user documentation explaining logging in, password resets, and accessing resources.
- Verify that users and groups are created and configured correctly within the FreeIPA domain.
- If using Kerberos, test user authentication using Kerberos tickets to ensure functionality.
- Verify that client software on domain members is functioning correctly and users can access resources

Installing Standalone Kerberos Server (no FreeIPA)

sudo dnf install krb5-server (client is krb5-workstation, krb5-libs, krb5-user)

/etc/krb5.conf: Main configuration file; defines Kerberos realm, KDC locations, encryption types, etc.
/etc/krb5/kdc.conf: Configuration for the KDC (server-side) if you're setting up a Kerberos server.
/etc/krb5/login.conf: Defines how Kerberos is used for authentication (login).
/var/lib/krb5/krb5.keytab: Stores the master Kerberos key for the KDC.

Client-side commands:
kinit <principal> - For client machine users to get a ticket to access a Kerberos-protected service.
klist -f - Lists all available Kerberos tickets held by the user, for verifying and seeing lifetime, -f gives more info
kdestroy - Destroys a specific Kerberos ticket. For logging out of a service or freeing up resources.
Server-side commands:
kdb5_util - Manages the Kerberos database, keytabs, and principals
kadmin.local - Manages Kerberos principals and credentials: creating, modifying user accounts, resetting passwords, and managing keytabs used by the KDC.

Mentionable related commands/ items:
keyutils - general-purpose tool for managing keyrings and keys, manage Kerberos keytabs alongside other key management tasks. (it itself doesn't interact directly with the Kerberos database)
sshd_krb5_module: This isn't a standalone command, but rather a module used by the SSH daemon to enable Kerberos authentication for SSH connections. You can configure it through SSH configuration files.

Systemd Services:
krb5-kdc.service (server-side): Manages the KDC daemon.
krb5-kadmind.service (server-side): Manages the Kerberos administration daemon.

TCP/88 (default): messages between clients and KDC. TCP is more secure but Windows clients may need UDP

Important Configurations:
Realm: Unique identifier for your Kerberos domain (e.g., EXAMPLE.COM).
KDC Locations: Specify the hostname or IP address of your KDC servers.
Default Encryption Type: Choose an appropriate encryption type (e.g., aes256-cts).
Ticket Lifetime: Set the expiration time for Kerberos tickets.
Client Principal: Define the principal name for your client machine (e.g., host/hostname`).

Managing the Kerberos database, keytabs, and principals with kdb5_util

Create and initialize database and set master password	kdb5_util create -r <realm> -s <keytab_file> -P <passwd>
Create new principal in <realm> with new <password>	kdb5_util addprinc -r <realm> -p <password> <principal>
Modifiy existing principal's attributes (e.g., password, flags)	kdb5_util modifyprinc -r <realm> <principal>
Removes a principal from DB	kdb5_util deleteprinc -r <realm> <principal>
Lists all principals in <realm> with key versions (-kv)	kdb5_util listprinc -r <realm> -kv
Create Keytab	kdb5_util create -r <realm> -s <keytab_file>
Add Entries to Keytab	kdb5_util addprinc -r <realm> -p <passwd> -t <keytab_file> <principal>
Merge Keytabs	kdb5_util merge -s <target_ktab> <source_ktab1> <source_ktab2> ...
Dump Database (can expose sensitive information)	kdb5_util dump -r <realm> -f <output_file>
Verify integrity of the Kerberos database	kdb5_util verify -r <realm>

Manage Kerberos with kadmin.local
Running the command kadmin.local alone will drop you into it's own CLI

Create a new principal for KDC administration:
addprinc -randkey kdc_admin@EXAMPLE.COM
The -randkey option is to generate a random password; kdc_admin@EXAMPLE.COM to name the principal and EXAMPLE.COM representing the Kerberos realm name.

Exit kadmin.local by entering quit.

Grant the kdc_admin principal the permissions to manage the KDC:
kadmin.local -p krb5/admin@EXAMPLE.COM ktadd -k /etc/krb5.keytab kdc_admin@EXAMPLE.COM

The first part "-p krb5/admin@EXAMPLE.COM" provides the password for the krb5/admin principal (usually the root principal) that has full administrative privileges in the Kerberos database.
The second part "ktadd..." adds the key for the kdc_admin principal to the specified keytab file (/etc/krb5.keytab)

Restrict access to kadmin.local using the /etc/sudoers file:
Run "nano /etc/sudoers" and add a block like this:

```
# Allow users in the 'kdc_admin' group to run kadmin.local as kdc_admin@EXAMPLE.COM
%kdc_admin ALL = NOPASSWD: /usr/sbin/kadmin.local -p kdc_admin@EXAMPLE.COM
```

"%kdc_admin" sets the rule applies to users in the kdc_admin group (create it using "groupadd kdc_admin")
"ALL = NOPASSWD" allows group members to run kadmin.local without a password, but only when using the kdc_admin@EXAMPLE.COM principal using the -p option.
"/usr/sbin/kadmin.local -p ..." simply specifies the command with sudo privileges.

Verification:
Create a user account that belongs to the kdc_admin group you created, og in as the newly created user.
Run "sudo kadmin.local -p kdc_admin@EXAMPLE.COM"
You should be prompted for the password of the kdc_admin principal (the one generated in step 2). If successful, you'll enter kadmin.local mode impersonating the kdc_admin principal.

Other Kadmin commands

addprinc <principal>	Adds a new principal (user or service account) to the database
delprinc <principal>	Deletes a principal from the Kerberos database
modprinc <principal>	Modifies attributes of an existing principal
rename_principal <old> <new>	Renames an existing principal in the Kerberos database
change_password <principal>	Changes the password of an existing principal
cpw <principal>	Alias for change_password
listprincs	Lists all principals in the Kerberos database
getprinc <principal>	Retrieves and displays information about a specified principal
ktadd -k <keytab_file> <principal>	Adds a principal's key to a keytab file (for passwordless authentication)
ktremove -k <keytab_file> <principal>	Removes a principal's key from a keytab file
ktdestroy -k <keytab_file>	Destroys a keytab file (use with caution)
getprivs	Shows administrative privileges of current user for kadmin.local CLI
listpols	Lists all policies in database (password rules, ticket lifetimes, etc.)
addpol <policy>	Adds a new policy to the Kerberos database.
modpol <policy>	Modifies attributes of an existing policy.
delpol <policy>	Deletes a policy from the Kerberos database.
getpol <policy>	Retrieves and displays information about a specified policy.
purgekeys <principal>	Removes all keys for a principal that are not the most recent.

SELinux Booleans

allow_httpd_pkey_init	Needed if using HTTP for key distribution.
allow_kadmind_port	Access on TCP 464 for administrative access to the KDC.
allow_kerberos_dce	Needed to support DCE clients using Kerberos.
allow_kerberos_kdc_tcp_port	Enables TCP traffic for the KDC
allow_kerberos_tgt_deleg	Enables delegation of Ticket-Granting Tickets (TGTs)
allow_mit_krb5_migrate	Needed if migrating existing Kerberos principals.
allow_smbd_krb5_right	Required if using Kerberos for Samba authentication.
allow_sshd_klogin	Enables Kerberos login for SSH connections.
allow_unreserved_ports	Allow applications to bind to privileged ports (ports 1-1024)

SELinux File Contexts

/etc/krb5.conf	etc_krb5_conf_t
/var/lib/krb5	var_lib_krb5_t
/var/log/krb5	var_log_krb5_t
Keytab - /etc/krb5.keytab)	krb5_keytab_t
/usr/sbin/kadmin, /usr/sbin/krb5kdc	usr_sbin_krb5_t
/run/krb5 (if used)	var_run_krb5_t

```
firewall-cmd --permanent --add-service=krb5 # Opens default Kerberos ports (TCP 88 and UDP 88)
firewall-cmd --permanent --add-service=kadmind # Opens KDC administration port (TCP 464)

iptables -A INPUT -p tcp --dport 88 -j ACCEPT
iptables -A INPUT -p udp --dport 88 -j ACCEPT
iptables -A INPUT -p tcp --dport 464 -j ACCEPT
systemctl restart krb5kdc kadmin
```

Client configuration - /etc/krb5.conf

```
[libdefaults]
    default_realm = EXAMPLE.COM
    ticket_lifetime = 24h
    renew_lifetime = 7d
[realms]
    EXAMPLE.COM = {
        kdc = kerberos.example.com
        # Optional: Specify additional KDC servers for redundancy
        # kdc = kerberos1.example.com
        # kdc = kerberos2.example.com
    }
[domain_realm]
    .example.com = EXAMPLE.COM
```

Server configuration example /etc/krb5/kdc.conf

```
[kdcdefaults]
    # Define encryption types supported by the KDC
    permitted_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96
    default_keytab = /etc/krb5/kdc.keytab
[realms]
    EXAMPLE.COM = {
        # Master key location (use kdb5_passwd to create)
        master_key_file = /var/lib/kerberos/krb5.keytab
        # Database for storing Kerberos principals (replace with your chosen database)
        database_module = kadm5
        # Database specific options
        database_name = EXAMPLE.COM  # Database name for the realm
        # Comment out if database resides on another machine (NOT good to have exposed on the network- don't!)
        # database_server = 192.168.1.10  # Replace with server IP (Not smart! See above)
        # admin_server = kerberos.example.com
        # Restrict access to the KDC based on IP address (Administrative Access Controls are a better option)
        # access_control = {
        #     host = 192.168.1.0/24  # Allow access from this subnet only
        # }
    }
```

NFS- Network File System
- Originally developed by Sun, access to remote file systems with mount points. NFSv4 listens on port 2049

	NFSv3	**NFSv4**
Packages	nfs-utils and nfs-utils-lib	nfs-common, nfs4-acl, rpc-svc-gss, for client. For server, use nfs-kernel-server and nfs-utils
Config files	/etc/sysconfig/nfs, /etc/exports	/etc/exports, /etc/nfs/, /etc/nfs/nfs.conf (server)
Services (SystemD & init)	/etc/init.d/nfs, /etc/init.d/rpcbind service nfs start, service rpcbind start	nfs-server.service and nfs-kernel-server.service, rpc-svc-gss.service
Permissions	/etc/exports file, some options given	ACLs and more options added in /etc/exports
RPC	rpc.lockd, rpc.statd (file locking, status info)	Now automated/ dynamic in NFS4 protocol
Automount	/etc/fstab/	/etc/fstab/ and autofs

NFSv3 Common permission options in the /etc/exports file
Self-explanitory: read-write (rw), read-only (ro), and no access (-)

root_squash	Forces all NFS requests from the root user (UID 0) on the client machine to be mapped to the anonymous user (usually nfsnobody) on the server, to help prevent accidental or malicious modifications by the root user on the client. [also "chown nfsnobody /share" for shared folder]
no_root_squash	Provides access with full permissions (if granted in the ACL). Use this option with caution due to potential security risks.
all_squash	A more aggressive version of root_squash that maps all client UIDs to the anonymous user on the server, regardless of the client user (has significant security implications).
anonuid/anongid	Allow you to specify the UID and GID of the anonymous user on the server used for mapping client users when root_squash or all_squash is enabled.
sync/ async	These have the same functionality as used with the mount command

You can specify access rights for specific hosts or networks. Wildcards can be used to represent groups of hosts or networks in the access control list. you can also use CIDR notation like 192.168.1.0/24

NFSv4-introduced new features and functionalities
Added options for overriding inheritance and setting specific permissions for individual directories in /etc/exports
- Fake root mount: if a server exporting /home and /data, instead of mounting both, just mount /
- Allows for clients to send their UID/GID info for access control decisions
- Access Control Lists (ACLs) on exported directories- the nfs4-acl package provides extensions for setfacl and getfacl to better support NFS, for more granular access permissions for specific users and groups on the server.
- Security options provide more granular authentication and authorization mechanisms like using a Kerberos server. An older option NLM server (Network Lock Manager) is supported but outdated, and NFS delegation tickets (built into NFS) has been proved to have vulnerabilities, thus Kerberos solutions seem to remain as the best security solution. Kerberos options are krb5p, krb5i, and none. Use krb5p for Kerberos security with encryption for data privacy and integrity protection. The krb5pi uses Kerberos but without data encryption. None is obviously not recommended. There was also a systemd sevrvice called nfs-secure-server.service but it integrated into nfs-server.service

Some sample entries in the NFSv4 /etc/exports file
```
/data *(rw,sync,all_squash)
        # Export /data directory with read-write access for all (only all_squash for basic security- this is weak and not advised)
/directory_to_export -sec=krb5p,rw,root_squash,sync  # Consider using ACLs instead of root_squash
        # Export with Kerberos, read-write access, root squashing, and sync
/home/users (rw,sync,nfsv4,user_acl,sec=krb5p)  # Preferred approach
        # Export /home/users with user ACLs, NFSv4, and Kerberos security
```

Commands:
Use exportfs to manage NFS exports. Options include -v to list currently exported directories; -a to export all entries in /etc/exports. To add new exports use 'exportfs -o options /dir client_IP' and to remove exports use 'exportfs -u /directory'
The command 'showmount -e' gives info on NFS shares currently exported; rpcinfo nfs gives info about the NFS server's RPC, and nfsstat gives statistics related to NFS server activity.

Client-side: mount, showmount -e server_IP (show NFS shares on a server), and nfsstat for NFS client statistics

The autofs package:
autofs.service: The systemd service that manages the autofs daemon itself. (start, stop, and restart)
automount: Command for manual interaction with autofs, manually mounting or unmounting specific automount points or debugging automount behavior (viewing logs or checking status of automount points)

/etc/auto.master: main configuration file - defines mount points, links to their map files, global options for autofs
Each line in the file typically follows this format: <mount_point> <map_file> [<options>]
/etc/auto.<identifier>: Map files, referenced by the master map file, contain details on individual automount points.
Each map file defines a specific automount configuration for a particular mount point. The format depends on the chosen map type:
amd.map format: Traditional format that specifies the NFS server location, automount behavior, access control.
auto.master.d format: Newer format that allows for inheritance and modular configuration based on directories.

NFSv3 Mount Options:
rsize/ wsize (read/write size): Set maximum packet size per request the client will try to read or write from the NFS server. Increasing can improve performance for large file reads, but values too large might lead to fragmentation and inefficient network usage.

bg/fg (background/foreground): background allows the system to continue booting while the NFS mount is being established, while foreground forces the mount command to wait until the NFS mount is successfully completed before returning.

async/ sync: The default, asynch allows the client to acknowledge write requests to the server before data is physically written to disk. Can improve performance but might lead to data loss if the client crashes before the write is completed. Using sync ensures that all data written to the NFS share is flushed to the server's disk before the mount command returns. This can improve data integrity but can also impact performance.

noauto: prevents the system from automounting the NFS share during boot

_netdev: Tells system to wait for the network interface to be configured before attempting the NFS mount

tcp/udp: NFSv3 typically defaults to TCP, but UDP can be used in specific scenarios (like low-latency networks) with trade-offs in reliability.

NFSv4 Mount Options:
Building on NFSv3 options, NFSv4 introduces additional options related to security flavors and performance optimizations:

sec: krb5p/ krb5i/ none: Use krb5p for Kerberos security with encryption for data privacy and integrity protection. The krb5pi uses Kerberos but without data encryption. None is obviously not recommended. Kerberos needs to be separately configured on the client and server. rpc_sec is a remnant of NFSv2 and no longer relevant.

nfsvers (NFS version): While the system might negotiate the NFS version with the server, you can explicitly specify nfsvers=4 to force an NFSv4 mount.

minor_version (minor NFS version): This option allows specifying a specific minor version of NFSv4 if your server supports multiple versions.

readdirplus: This option enables the client to request additional information along with directory listings, potentially improving performance for browsing directories on the NFS share.

Many NFSv3 mount options like rsize, wsize, sync/async, noauto, background/foreground and _netdev are still relevant for performance tuning and basic mount behavior in NFSv4.

Kerberos setup

Log in to the kerberos server	ssh <username>@<kerberos_server_ip>
Launch kadmin.local to enter CLI	sudo kadmin.local
Add the service principal	addprinc -randkey nfs/<nfs_server_hostname>
Create a keytab file for NFS Server	ktadd -k /etc/krb5.keytab nfs/<nfs_hostname> (hit return, type quit to exit CLI)
Copy file to NFS Server	scp /etc/krb5.keytab <username>@<nfs_server_ip>:/etc/krb5.keytab (hit return, then exit ssh)
Set write permissions for keytab file	sudo chmod 400 /etc/krb5.keytab; sudo chown root:root /etc/krb5.keytab
Be sure /etc/exports file has shares fixed	sec=krb5p or similar, as demonstrated before
Edit /etc/sysconfig/nfs (enable GSS/Kerb)	sudo vi /etc/sysconfig/nfs Add or uncomment lines RPCGSSDARGS="" and RPCSVCGSSDARGS=""
Edit /etc/krb5.conf to check config	sudo vi /etc/krb5.conf - Ensure [realms], [domain_realm] are configured
Enable, start NFS and kerberos	sudo systemctl enable nfs-server rpcbind && sudo systemctl start nfs-server rpcbind && sudo systemctl restart nfs-server
Run on client to verify it's working	sudo mount -t nfs -o sec=krb5p $server_hostname:/$share_name /mnt/nfs

firewalld configuration: sudo firewall-cmd --permanent --add-service=nfs
iptables configuration: sudo iptables -I INPUT -p tcp --dport 2049,111 -j ACCEPT
List SELinux stuff with semanage boolean -l | grep -i '(nfs_)'; semanage fcontext -l | grep -i '(nfs_)'

Samba - SMB

Client executables (install package: samba-client)

- smbcontrol — Manage Samba shares, view connections to servers), and perform client administrative tasks
- smbclient — Browse, copy, manage files/directories on remote Samba servers.
- smbmount — Mount remote Samba shares as local directories.
- nmblookup — Perform NetBIOS name resolution (find Samba servers on network).
- smbcacls — Remote management tool- view and modify Windows ACLs on files hosted on a Samba server.

SystemD services on the client

- smbd.service — Main SMB/CIFS daemon, handles file/print services.
- nmbd.service — Provides NetBIOS name service.
- cifs.service — Mounts CIFS/SMB shares

Server executables (install package: samba)

- samba-tool — For managing configuration, users, shares, Kerberos settings, and other administrative tasks
- smbstatus — Shows the current status of the Samba server, including active connections and shares.
- testparm — Verifies the syntax of your /etc/samba/smb.conf file before restarting the service.
- smbpasswd — Changes Samba user account passwords.

SystemD services on the server (also includes the client list)

- smbd.service — Main SMB/CIFS daemon, providing access to clients, handles file/print services.
- nmbd.service — Provides NetBIOS name service.
- samba.service — Alias for smbd.service.
- winbind.service — Allows Windows domain authentication.
- samba-ad-dc.service — Samba Active Directory Domain Controller service

Main config file is /etc/samba/smb.conf (resources, user authentication, security, etc.)
Optional configs: /etc/samba/smb-security.conf (security settings) or /etc/samba/smb.secrets (for sensitive info)

/etc/samba/smb.conf - specify shared directories or files; access permissions for users and/or groups; LDAP and encryption options, and security settings

- Shares: Define shared directories or files using the [sharename] section.
- Permissions: read only = yes, writeable = yes, and specific user/group entries.
- LDAP Integration: security = ads option and specifying LDAP server details.
- Security Settings: Enhance security with options like:
  - encrypt passwords = yes - Encrypts user passwords during storage.
  - map to guest = bad user - Disables guest access.
  - browsable = no - Hides the share from browse lists.
  - valid users = @users - Restricts access to the local usernames or users in group specified.

Example smb.conf:

```
[Global]
  workgroup = MYWORKGROUP  # Name of your workgroup for network browsing
  server string = My Samba Server  # Descriptive name for your Samba server
  security = ads  # Enable LDAP security for user authentication
  encrypt passwords = yes  # Encrypt user passwords for added security
  map to guest = bad user # No access for the unauthenticated- gives guest access as 'bad user' which doesn't exist (denied!)
  # wins server = 10.0.0.1  # WINS server IP for name resolution, logging, and specifying one interface to listen (not all)
  logging = file  # Enable file-based logging
  log level = warn  # Log warnings and more severe messages
  log file = /var/log/samba/smb.log  # Specify the log file location
  # interfaces = 192.168.1.0/24  # Example: Listen only on the 192.168.1.0/24 subnet
[SharedFolder]
  path = /home/share  # Path to the directory you want to share
  browsable = no  # Hide this share from network browsing
  writable = yes  # Allow users with access to modify files
  read only = no  # Allow both reading and writing
  valid users = @share_access  # Grant access only to users in the "share_access" group
  create mask = 0664  # Example: New files get rw-rw---- permissions (user:group:others)
  directory mask = 0775  # Example: New directories get rwxrwxr-x permissions
[AnotherShare]
  path = /var/www/html  # Path to the web server's document root (example)
  read only = yes  # Allow users to only read files in this share
  valid users = user1 user2 @web_admins  # Grant access to specific users and a group
  # locking = ...  # Options for controlling how multiple clients share access to files
  # oplocks = ...  # Options related to optimistic locking (advanced)
  # write cache (or 'read cache') = yes  # Enable caching writes/ reads for faster performance
```

The "map to guest = bad user" matches any user that fails authentication (denied)

Kerberos implementation

Log in to the Kerberos server	ssh <username>@<kerberos_server_ip>
Launch kadmin.local to enter CLI	sudo kadmin.local
Add the service principal	addprinc -randkey smb/<smb_server_hostname>
Create a keytab file for SMB server	ktadd -k /etc/krb5.keytab smb/<smb_hostname> (hit return, type quit to exit CLI)
Copy file to SMB Server	scp /etc/krb5.keytab <username>@<smb_server_ip>:/etc/krb5.keytab (hit return, then exit ssh)
Set write permissions for keytab file	sudo chmod 400 /etc/krb5.keytab; sudo chown root:root /etc/krb5.keytab
Edit /etc/krb5.conf to check config	sudo vi /etc/krb5.conf - Ensure [realms], [domain_realm] are configured
Ensure /etc/samba/krb5users is ready	The file will need entries similar to example provided below
Enable, start SMB and kerberos	sudo systemctl enable smb rpcbind && sudo systemctl restart smb
Run on client to verify it's working	sudo mount -t cifs -o sec=krb5i,username=<client_username>@<REALM> //$server_hostname/$share_name /mnt/smbshare

Create a user mapping file to translate Kerberos principals (username@REALM) to specific Samba usernames

```
# /etc/samba/krb5users
# Map Kerberos principal "user1@MYREALM" to Samba user "samba_user1"
user1@MYREALM = samba_user1   # samba_user1 would be the username on the Linux host
```

```
[Global]
  workgroup = MYWORKGROUP
  server string = My Samba Server
  security = krb5i  # Enable Kerberos integration with user mapping
  encrypt passwords = yes
  map to guest = bad user
  username map = /etc/samba/krb5users  # Specify the user mapping file location
[SharedFolder]
  path = /home/share
  browsable = no
  writable = yes
  read only = no
  # Allow access only to users mapped in the user mapping file
#  valid users = %    using % means all Kerberos users logged in!
  valid users =  samba_user1
```

More security stuff:

The file /etc/samba/smb.secrets stores encrypted passwords (machine, keys). Use alternative approaches within /etc/samba/smb.conf first. If using Kerberos keytabs, this file can be emptied.

firewalld configuration: sudo firewall-cmd --permanent --add-service=samba
iptables configuration: sudo iptables -I INPUT -p tcp --dport 137-139,445 -j ACCEPT
List SELinux stuff with semanage fcontext -l | grep -i '(smb_|samba_)' semanage boolean -l | grep -i '(smb_|samba_)'

Extending SMB: The "Samba-VFS" framework

Is not a virtual file system as named. Leverage to enhance SMB server with modules, shared libraries (.so files- think ldd, not kernel libraries like .ko) and they can be declared in /etc/samba/smb.conf using the vfs objects parameter. Some modules can be chained, allowing multiple modules to work sequentially.

```
[global]
  vfs objects = full_audit  # Load the vfs_full_audit module-  detailed logs of file operations for enhanced security
  vfs_full_audit_log_dir = /var/log/samba/audit  # Specify log directory
  vfs_full_audit_log_file = full_audit.log    # Specify log filename
  vfs_full_audit_log_rotate = 5            # Rotate logs after 5 rotations
  vfs_full_audit_log_size = 10M              # Maximum log size (10 Megabytes)
```

```
[global]
  vfs objects = acl_tdb  #  Access control through storage of Access Control Lists (ACLs)
  vfs_acl_tdb_path = /etc/samba/acl.tdb     # Specify TDB database path.
```

vfs_recycle (recycle bin to recover deleted files); vfs_usershare (give users share definitions); vfs_fruit: (Apple File System (AFP) macOS client shares); vfs_fake_chroot (make a chroot environment for each connected user); vfs_deny_hosts (restrict access to shares based on IP or hostnames); vfs_cifs_xattr (enables storing extended attributes on Samba shares)