21 Clientanbindung

Die bisherigen Linux-Beispiele dieses Buches haben bislang stets den `root`*-Account verwendet. In den folgenden Kapiteln benötigen Sie jedoch zentral verwaltete Betriebssystem-Accounts ohne spezielle Privilegien. Daher behandelt das vorliegende Kapitel die Anbindung der Windows- und Linux-Benutzerverwaltung an die geschaffene Infrastruktur. Die LDAP-Dienste aus den Active-Directory-Umgebungen sowie auch die OpenLDAP-Verzeichnisse auf Basis des LDAP-Kapitels 20 spielen hier die tragende Rolle, da diese Verzeichnisse die notwendigen Benutzerattribute und Autorisierungsdaten liefern werden. Ergänzend übernimmt die Kerberos-Infrastruktur die Aufgabe der Passwortüberprüfung und dient außerdem der kryptografischen Sicherung der LDAP-Netzwerkdaten per SASL/GSS-API.*

Zuerst erfolgt in Abschnitt 21.1 die Anbindung von Windows-Clients an die Active-Directory-Teilstruktur unterhalb `ADS.EXAMPLE.COM`*. Dieser relativ triviale Schritt wird hier nur kurz angerissen. Eine vollwertige Anbindung von Windows-Clients an die eigenständige Kerberos- und LDAP-Infrastruktur ist nicht möglich, da Windows-Clients daneben noch weitere Infrastrukturkomponenten benötigen, die nur durch Active Directory – oder eine dazu kompatible Infrastruktur wie Samba 4 – angeboten werden.*

Danach werden in Abschnitt 21.4 die bisherigen LDAP-Daten in den Beispieldomänen der verteilten Gesamtstruktur dieses Buches um Linux-Benutzerdaten (sogenannte POSIX-Informationen) erweitert. Das Hauptaugenmerk der restlichen Abschnitte dieses Kapitels liegt auf der Integration von Linux-Systemen in die heterogene Gesamtstruktur. Am Ende dieses Kapitels können sich dann Nutzer aus den verschiedenen Realms an Clientmaschinen innerhalb der Gesamtstruktur anmelden.

21.1 Windows-Clients in Active Directory

Unter dem Begriff »*Clientanbindung*« soll hier die Anbindung des Betriebssystems an die zentrale Kerberos-/LDAP-Verwaltungsinfrastruktur

verstanden werden. Diese Aufgabe lässt sich grundsätzlich in folgende Teilaspekte gliedern:

1. Anbindung des Betriebssystems an die *Identitätsinformationen* und *Autorisierungsdaten* der zentralen Infrastruktur. Dadurch »kennt« das Betriebssystem die Nutzer und deren Eigenschaften und kann beispielsweise auf Basis von Gruppenzugehörigkeiten Zugriffsentscheidungen treffen.
2. Anbindung der lokalen *Anmeldung* am Betriebssystem an den zentralen Authentisierungsdienst. Damit kann das Betriebssystem die Passwortüberprüfung über einen Authentisierungsdienst wie Kerberos abwickeln.
3. Sicherheit des gesamten Vorgangs. Das beinhaltet die Überprüfung der *Authentizität* und der *Vertrauenswürdigkeit* der zentralen Infrastruktur sowie der von ihr gelieferten Daten im Hinblick auf 1 und 2.

Windows: Domain Join

Windows-Domänen binden einen Client beim Beitritt eines Systems in die AD-Domäne ein. Dieser sogenannte *Domain Join* umfasst alle drei Aspekte. Wie er bei Windows 10 durchgeführt werden kann, wird im Folgenden gezeigt.

Der Name des neuen Systems soll `win01.ads.example.com` lauten. Das Ziel ist der Beitritt zur AD-Domäne `ADS.EXAMPLE.COM`. Folgende Schritte sollten Sie im Voraus ausführen:

- Installieren Sie das Betriebssystem Windows 10.
- Setzen Sie den Hostnamen auf `win01`.
- Konfigurieren Sie den DNS-Client, sodass er den DNS-Dienst aus Abschnitt 15.2 (Seite 307) verwendet.

Sind diese Voraussetzungen erfüllt, so können Sie den Domain Join wie folgt durchführen:

- Melden Sie sich auf der `win01` unter einem Account mit lokalen Admin-Berechtigungen an.
- Klicken Sie in der App *Dieser PC* mit der rechten Maustaste auf *Eigenschaften* und wählen Sie den Punkt *Einstellungen ändern*.
- Darauf öffnet sich das Fenster *Systemeigenschaften*. Klicken Sie darin auf *Ändern…*
- In dem Fenster *Ändern des Computernamens bzw. der Domäne*, das sich daraufhin öffnet, markieren Sie *Domäne*.
- Geben Sie dort `ADS.EXAMPLE.COM` ein und klicken Sie *OK* (Abbildung 21.1).

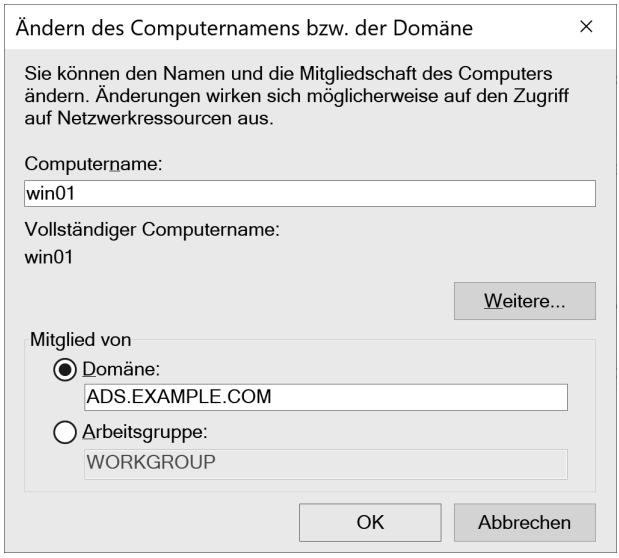

Abb. 21.1
Auswahl der Domäne beim Join

- Im nächsten Fenster mit dem Titel *Windows-Sicherheit* (Abbildung 21.2) geben Sie die Anmeldeinformationen eines berechtigten Accounts in der Domäne an. Das ist in der vorliegenden Testumgebung der Domänen-Account `Administrator`.

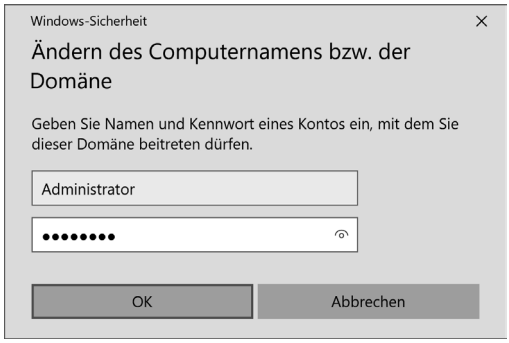

Abb. 21.2
Angabe eines berechtigten Domänen-Accounts

- Als kleines Erfolgserlebnis sollte daraufhin die Meldung aus Abbildung 21.3 zu sehen sein. Danach steht als letzter Schritt noch ein Reboot des Systems an.

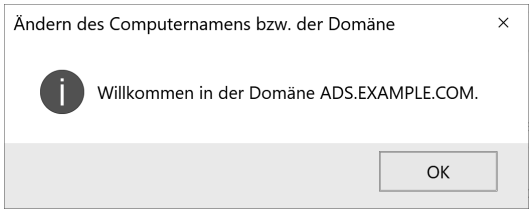

Abb. 21.3
Erfolgreicher Domain Join

Nach dem Neustart der win01 haben Sie die Möglichkeit, sich mit einem der Test-Accounts aus Kapitel 15 anzumelden. Das System hat durch den Join neben seiner eigenen lokalen Benutzerdatenbank auch Zugriff auf die domänenweite Benutzerdatenbank der AD-Domäne ADS.EXAMPLE.COM, es kennt also die zentral verwalteten Nutzer und kann deren Passwörter überprüfen. Das gilt übrigens auch für die Nutzer der beiden Subdomänen MYDOM.ADS.EXAMPLE.COM und OTHERDOM.ADS.EXAMPLE.COM. Windows-Clients erhalten durch den Domain Join Zugriff auf die Benutzerdatenbanken sämtlicher Domänen eines Active Directory Forest.

Für die Sicherheit sorgt der neue Maschinen-Account mit dem Namen WIN01$, der im Rahmen des Domain Join in der ADS-Domäne angelegt worden ist. Dessen Passwort und die daraus abgeleiteten kryptografischen Schlüssel stellen die Kerberos-Schlüssel der win01 dar.

Bei Windows gibt es hier also nicht viel zu tun. Doch auch Linux steht dem inzwischen in nichts mehr nach, wie der folgende Abschnitt zeigt.

21.2 Linux-Clients in Active Directory

Die Integration von Linux-Clients war in den Anfangsjahren von Active Directory noch eine Herausforderung, die zahllose Buchseiten füllen konnte – obwohl die Windows-Benutzerverwaltung auf den in der Unix-Welt längst etablierten Protokollen Kerberos und LDAP aufsetzte. Doch abweichende Standards und andere subtile Inkompatibilitäten machten den Administratoren das Leben schwer. Inzwischen haben beide Seiten sich zusammengerauft. Active Directory etwa kann ab Werk mit spezifischen Attributen für die Linux-Welt umgehen, und aktuelle Linux-Distributionen haben umgekehrt bereits alle nötigen Werkzeuge an Bord, um sich bequem in Microsofts Benutzerverwaltung zu integrieren. Unter CentOS 8 können Sie das auf Ihrer Linux-Maschine lx01.ads.example.com beispielsweise mit dem Kommando realm nachvollziehen. Stellen Sie zunächst per dnf install realmd sicher, dass die nötige Software vorhanden ist. Installieren Sie für erste Tests außerdem die Pakete krb5-workstation und openldap-clients. Ansonsten benötigen Sie noch dieselben Informationen wie schon bei Domain Join der win01, nämlich den Namen der Domäne, der Sie beitreten möchten, sowie Name und Passwort eines zugehörigen administrativen Benutzers. Listing 21.1 zeigt den vollständigen Aufruf.

Software: – realmd

Listing 21.1 *Domänenbeitritt eines Linux-Systems auf der Schnellspur per* realmd

```
root@lx01.ads:~# realm join                                    \
                        -U Administrator ADS.EXAMPLE.COM
Password for Administrator: P@ssw0rd
root@lx01.ads:~#
```

Die Maschine `lx01.ads` ist damit Teil der Domäne `ADS.EXAMPLE.COM`. Anders als bei Windows ist noch nicht einmal ein Neustart des Systems nötig.

Sofort im Anschluss existieren Windows-Benutzer aus Active Directory nun auch unter Linux, besitzen dort ihre gewohnten Gruppenmitgliedschaften und erhalten beim Login ein Kerberos Ticket, um per Single Sign-on auf weitere Dienste zugreifen zu können. Listing 21.2 demonstriert das am Beispiel eines SSH-Logins für den Windows-Benutzer `Administrator`, der danach per Kerberos auf den LDAP-Dienst des Domain Controller zugreift.

```
root@lx01.ads:~# ssh -l administrator@ads.example.com      \
                      lx01.ads.example.com
administrator@ads.example.com@lx01.ads's password: P@ssw0rd
[administrator@ads.example.com@lx01 ~]$ ldapwhoami         \
                      -Y GSSAPI                            \
                      -H ldap://kdc01.ads.example.com
SASL/GSSAPI authentication started
SASL username: Administrator@ADS.EXAMPLE.COM
SASL SSF: 256
SASL data security layer installed.
u:ADS\Administrator
[administrator@ads.example.com@lx01 ~]$
```

Listing 21.2 *Windows-Benutzer können das Linux-System unmittelbar verwenden und SSO-Fähigkeiten nutzen.*

In Active Directory wurde ein Maschinenobjekt mit zufällig erzeugtem Kennwort angelegt. Davon abgeleitete Kerberos-Schlüssel für verschiedene Principals sind in der Datei `/etc/krb5.keytab` abgelegt (siehe Listing 21.3). Unter der Haube nutzt `realmd` dazu das bereits bekannte Werkzeug `adcli`.

```
root@lx01.ads:~# klist -k
Keytab name: FILE:/etc/krb5.keytab
KVNO Principal
---- --------------------------------------------------
   2 LX01$@ADS.EXAMPLE.COM
   2 LX01$@ADS.EXAMPLE.COM
   2 host/LX01@ADS.EXAMPLE.COM
   2 host/LX01@ADS.EXAMPLE.COM
   2 host/lx01.ads.example.com@ADS.EXAMPLE.COM
   2 host/lx01.ads.example.com@ADS.EXAMPLE.COM
   2 RestrictedKrbHost/LX01@ADS.EXAMPLE.COM
   2 RestrictedKrbHost/LX01@ADS.EXAMPLE.COM
   2 RestrictedKrbHost/lx01.ads.example.com@ADS.EXAMPLE.
➥  COM
```

Listing 21.3 *Vom `realmd` erzeugte Kerberos-Keytab mit Host Credentials*

```
   2 RestrictedKrbHost/lx01.ads.example.com@ADS.EXAMPLE.
➥  COM
```

Hierbei fällt auf, dass die Anbindung augenscheinlich nicht nur eine, sondern mehrere Domänen unterstützt. Damit alle Benutzer- und Gruppennamen trotzdem eindeutig bleiben, werden sie um den Domänennamen ergänzt. Zudem kümmert `realmd` sich offenkundig um mehr als die reine Benutzerverwaltung und konfiguriert auch noch sinnvolle Komfortfunktionen im System. Normalerweise würde man etwa beim SSH-Login in Listing 21.2 einen Fehler erwarten, weil das Homeverzeichnis für den bislang unbekannten Benutzer `administrator@ads.example.com` fehlt. Tatsächlich aber passt `realmd` das System so an, dass fehlende Homeverzeichnisse automatisch neu angelegt werden.

Ein Blick mit `realm --help` auf die Liste der unterstützten Subkommandos und Optionen zeigt eine erstaunlich übersichtliche Liste für eine komplexe Aufgabe wie die Verwaltung von Domänenmitgliedschaften. Zwar erlaubt `realmd` über Konfigurationsdateien noch etliche Anpassungsmöglichkeiten mehr. Doch letztlich weisen diese Punkte darauf hin, dass es sich bei `realmd` lediglich um eine Zusatzschicht handelt, die den Domänenbeitritt möglichst einfach und bequem halten soll. Die Hauptarbeit erledigt ein anderer Dienst, dem sich die nächsten Abschnitte eingehend widmen.

21.3 Der System Security Services Daemon

System Security Services Daemon

Dreh- und Angelpunkt bei der Anbindung moderner Linux-Systeme an eine zentrale Benutzerverwaltung ist der System Security Services Daemon, kurz `sssd`. Er bildet einerseits die Schnittstelle vom System nach außen zu Verzeichnis- oder Authentisierungsdiensten, interagiert mit lokalen Subsystemen von der Nutzerverwaltung bis zur Kerberos-Bibliothek, implementiert Cache-Funktionen für häufig benötigte Daten und stellt noch etliche optionale Zusatzfunktionen bereit bis hin zum eigenen Management-Dienst für Credential Caches. Der `sssd` ist dabei kein einheitlicher Dienst, sondern selbst wieder modular aufgebaut und erweiterbar. Entsprechend rasch wächst sein Funktionsumfang mit jeder neuen Version. So erschließt sich dann auch der Mehrwert des vergleichsweise einfachen und übersichtlichen `realmd`, denn auf den `sssd` treffen diese Attribute sicher nicht zu. Im Folgenden werden Sie daher lediglich die wichtigsten Eigenschaften und Konfigurationsmöglichkeiten des `sssd` kennenlernen, die im Rahmen der betrachteten Kerberos-Umgebungen eine Rolle spielen. Eine umfassende Beschreibung würde eigene Bücher füllen. Normalerweise erwartet Sie hier ein Verweis auf die Man Page mit detaillierten Informationen. Bei `sssd` handelt es sich ein einen gan-

zen Zoo von teils sehr umfangreichen Seiten für den Kerndienst, dessen Konfigurationsdatei sowie die einzelnen Module. Erfahrungsgemäß sind jedoch selbst diese Datenmengen noch nicht vollständig. Manche sonst undokumentierte Option ist mitunter nur in den Benutzerhandbüchern zu Red Hats Enterprise Linux beschrieben.

Achtung

Auch wenn das Buch primär auf Linux ausgerichtet ist, lassen sich viele der Beschreibungen und Konzepte auch unmittelbar auf andere Unix-artige Systeme übertragen. An dieser Stelle ist das anders. Einen zentralen Dienst mit dem breiten Funktionsumfang des `sssd` gibt es so nur unter Linux. Zum Ende des Kapitel werden Sie noch die Infrastruktur von PAM und NSS kennenlernen, die auch anderswo in der Unix-Welt verbreitet ist.

21.3.1 Einfache sssd-Konfiguration für Active Directory

Um zu verstehen, was beim Domänenbeitritt genau passiert, ist die Datei `/etc/sssd/sssd.conf`[1] ein wichtiger Ausgangspunkt. Listing 21.4 zeigt, mit welcher Konfiguration `sssd` im Anschluss an das Kommando `realm join` startet.

Listing 21.4 *Von* `realmd` *erzeugte* `sssd.conf`

```
[sssd]
domains = ads.example.com
config_file_version = 2
services = nss, pam

[domain/ads.example.com]
ad_domain = ads.example.com
krb5_realm = ADS.EXAMPLE.COM
realmd_tags = manages-system joined-with-adcli
cache_credentials = True
id_provider = ad
krb5_store_password_if_offline = True
default_shell = /bin/bash
ldap_id_mapping = True
use_fully_qualified_names = True
fallback_homedir = /home/%u@%d
access_provider = ad
```

[1] Die Laufzeitkonfiguration verwaltet `sssd` übrigens dynamisch in einer Datenbank. Wer neugierig ist, kann mit dem Kommando `ldbsearch tdb:///var/lib/sss/db/config.ldb` unter die Haube blicken.

Grundsätzlich kann sssd Mitgliedschaften in mehreren Domänen verwalten und lässt dabei für jede Domäne unterschiedliche Konfigurationsmöglichkeiten zu. Der Parameter domains im allgemeinen Abschnitt [sssd] verweist lediglich auf die spezifischen Konfigurationsabschnitte. Der gewählte Name ads.example.com muss dabei zur Benennung des Abschnitts [domain/ads.example.com] passen. Der Parameter services aktiviert Schnittstellen und Prozesse, die sssd an Standard-Subsysteme des Betriebssystems anbinden. Die folgenden Abschnitte zu NSS und PAM erläutern das im Detail.

Wichtigste Einstellung im Abschnitt [domain/ads.example.com] ist der Parameter id_provider. Er legt den Typ des Systems fest, das die Benutzerinformationen bereitstellt. Die verfügbaren Provider sind in Module aufgeteilt und in eigenen Man Pages genauer dokumentiert. Der Provider ad ist für die Integration in Active Directory zuständig und in der Man Page sssd-ad(5) detailliert beschrieben. Für FreeIPA gibt es entsprechend den Provider ipa. Der Provider ldap setzt keine integrierte Umgebung voraus und kann beispielsweise ein eigenständiges OpenLDAP-Verzeichnis ansprechen.

Neben dem id_provider für die Benutzerinformationen lassen sich noch separate Provider für diverse weitere Aufgaben konfigurieren. So legt auth_provider fest, wer für die Authentisierung von Benutzern zuständig ist, chpass_provider den Weg der Passwortänderungen, subdom_provider die Abfrage von vertrauenswürdigen Subdomänen in hierarchisch aufgebauten Umgebungen und noch einige weitere Provider für spezifische Anwendungsfälle. Häufig müssen diese Einstellungen nicht explizit gesetzt sein, denn sssd verwendet jeweils standardmäßig den id_provider. Den Zugriff zum lokalen System kann der access_provider steuern. Für ihn greift nicht der id_provider als Voreinstellung; vielmehr muss er bei Bedarf noch einmal explizit gesetzt werden.

Seine zugeordnete AD-Domäne entnimmt der ad-Provider dem Parameter ad_domain und kann daraufhin alle weiteren Informationen zur Umgebung über automatisierte DNS- und RPC-Abfragen ermitteln. Durch den explizit gesetzten Parameter krb5_realm verwendet sssd den Namen des zugehörigen Kerberos Realm in Großschreibung und entspricht so der unter Linux üblichen Konvention. In einem AD-Forest genügt es, die Wurzeldomäne zu konfigurieren. Die darunterliegenden Domänen werden automatisch erkannt und ebenfalls integriert.

Im Verzeichnis eines Active Directory stehen nicht notwendigerweise alle Informationen zur Verfügung, die ein Linux-System für einen Benutzer zwingend voraussetzt, beispielsweise die Login-Shell oder der Unix-Pfad zum Heimatverzeichnis. In der sssd-Konfiguration lassen sich deshalb mit default_shell und fallback_homedir Standardwerte vorgeben, falls im Benutzerobjekt kein spezifisches LDAP-Attribut vorhanden

ist. Windows wie Linux identifizieren Benutzer und Gruppen intern anhand von numerischen IDs, verwenden dafür allerdings unterschiedliche Formate. Ist der Parameter `ldap_id_mapping` aktiviert, berechnet `sssd` die sogenannten Posix-IDs für Linux über einen eindeutigen Algorithmus aus den jeweiligen Windows-SIDs. Ob Benutzer- und Gruppen unter Linux nur einen einfachen Namen erhalten oder wie unter Windows üblich auch ein Domänensuffix, legt die Einstellung `use_fully_qualified_names` fest. Im Endresultat können Active-Directory-Benutzer ein so eingebundenes Linux-System unmittelbar verwenden, ohne dass das Verzeichnis Linux-spezifische Benutzerattribute pflegen muss.

Hinter den Parametern `cache_credentials` und `krb5_store_password_if_offline` verbergen sich Komfortfunktionen des `sssd`, die erkennbar auf den Einsatz auf Mobilgeräten abzielen. Damit können sich bereits bekannte Benutzer auch ohne Netzverbindung zum Domain Controller mit ihrem letzten auf dem System verwendeten Passwort anmelden. Sobald die Verbindung wieder steht, beschafft ihnen `sssd` automatisch ein TGT.

Der `sssd` ist somit vollständig konfiguriert, um sich in die Benutzerverwaltung eines Active Directory zu integrieren. Darüber hinaus muss das Betriebssystem noch so modifiziert werden, dass es Benutzerinformationen über den `sssd` bezieht und den Dienst zur Authentisierung von Anmeldevorgängen verwendet. Auch diese Änderungen hat `realm join` bereits ausgeführt. Sie werden im Folgenden genauer beschrieben.

21.3.2 Name Service Switch (NSS) mit sssd

Die in Abschnitt 21.4.2 vorgestellten Dateien `/etc/passwd` und `/etc/group` stellen (zusammen mit der Datei `/etc/shadow`) die traditionelle Nutzerdatenbank eines Unix-Systems dar. Da es aber neben diesen lokalen Dateien weitere Quellen für Nutzer- und Gruppendaten gibt, unterstützen moderne Unix-/Linux-Betriebssysteme eine modulare Schnittstelle zu beliebigen weiteren Datenquellen. Dabei kann es sich wie im aktuellen Beispiel um einen lokalen Dienst wie `sssd` handeln oder um einen Netzwerkdienst wie NIS und LDAP. Der Name dieser Schnittstelle lautet *Name Service Switch (NSS)*. Für jede mögliche Datenquelle kann ein NSS-Modul konfiguriert werden, das POSIX-Daten aus dieser Datenquelle bezieht und sie dem System zur Verfügung stellt. Beispielsweise dient das Modul `/lib64/libnss_files.so.2` der Verwendung der Daten aus `/etc/passwd`, `/etc/shadow` und `/etc/group`, während `/lib64/libnss_sss.so.2` solche Daten via `sssd` beziehen kann. Für LDAP gibt es schließlich das Modul `/lib/libnss_ldap.so.2`.

`/etc/nsswitch.conf`

Welches NSS-Modul für welche Art von Namensinformation benutzt werden soll, wird in der Datei `/etc/nsswitch.conf` festgelegt. Nach einem

Bezeichner für die Art der Information kann man in der `nsswitch.conf` eine Liste von Modulen angeben. Der `realmd` hat beim Beitritt zur Domäne (indirekt über das Kommando `authselect`) das NSS-Modul `sss` unter anderem als primäre Quelle für `passwd`- und `group`-Informationen festgelegt (siehe Listing 21.5). Darüber hinaus ist keine weitere Konfiguration für das NSS-Modul selbst nötig. Lediglich `sssd` selbst muss wie oben gezeigt über den Eintrag `nss` im Parameter `services` so eingestellt sein, dass er die vom NSS-Modul verwendete Schnittstelle bereitstellt.

Listing 21.5 *Von* `realmd` *modifizierter Ausschnitt aus* `/etc/nsswitch.conf`

```
[...]
passwd:     sss files systemd
group:      sss files systemd
[...]
```

Auf diese Weise sind Benutzer aus Active Directory grundsätzlich im Linux-System bekannt. Damit auch das Passwort mithilfe von `sssd` und Kerberos gegen Active Directory geprüft wird, muss noch ein weiterer Bereich angepasst werden.

21.3.3 Pluggable Authentication Modules (PAM) mit sssd

Die *Pluggable Authentication Modules (PAM)* sind das Linux-Subsystem, das für Authentisierung und Zugriffskontrolle bei der Anmeldung zuständig ist. Darüber hinaus dient PAM auch der Vorbereitung der Anwendersitzung und kann Passwortänderungen steuern. Anmeldedienste wie beispielsweise die Linux-Anmeldung mit `login` oder die grafische Anmeldung mit dem *Gnome Display Manager* `gdm` müssen die verschiedenen Authentisierungsverfahren nicht selbst implementieren, sie können dazu einfach PAM benutzen.

PAM setzt sich aus einer Vielzahl einzelner Module zusammen, die die einzelnen Aspekte der Anmeldung konkret implementieren. Beispielsweise überprüft `pam_ldap.so` ein Passwort per LDAP-Simple-Bind, `pam_krb5.so`[2] nutzt Kerberos, `pam_sss.so` befragt dazu den `sssd`. Für die traditionelle Authentisierung und Autorisierung über Einträge in den Dateien `/etc/passwd` oder `/etc/shadow` (genauer eigentlich in den NSS-Maps `passwd` und `shadow`) ist das Modul `pam_unix.so` zuständig.

Konfigurationsdateien steuern das Zusammenspiel der einzelnen Module, wobei jede PAM-fähige Anwendung meist eine separate Konfigura-

[2] Ein früher häufig genutztes Modul, das in aktuellen RHEL- und CentOS-Distributionen nicht mehr enthalten ist. Gleichwertige Funktionen stellt heute der `sssd` bereit.

tion verwendet[3]. So kann etwa die PAM-Anmeldung über `sshd` anders konfiguriert sein als über `gdm`. Um die daraus resultierende Komplexität im Zaum zu halten, geben Linux-Distributionen häufig ein einfacheres Schema vor. Auf den CentOS-Systemen der Beispielumgebung liegen die gemeinsamen Vorgaben in `/etc/pam.d/system-auth` und einigen weiteren Dateien mit dem Suffix `-auth`. Von dort binden die anwendungsspezifischen Konfigurationsdateien ihre Einstellungen ganz oder teilweise ein. Als Hilfsmittel dient unter CentOS das Kommando `authselect`, mit dem sich auf einfache Weise feste Konfigurationsprofile verwalten lassen.

Beim Beitritt zur Domäne hat `authselect` neue Versionen der zentralen Dateien erzeugt, die für nichtlokale Benutzer die einzelnen Schritte der Anmeldung an `pam_sss.so` und damit an den `sssd` weiterleiten (siehe Listing 21.6). Der Eintrag `pam` im Parameter `services` der `sssd`-Konfiguration stellt sicher, dass die vom PAM-Modul verwendeten Schnittstellen bereit stehen.

```
auth required   pam_env.so
auth required   pam_faildelay.so delay=2000000
auth [default=1 ignore=ignore success=ok] pam_succeed_if.
➥  so uid >= 1000 quiet
auth [default=1 ignore=ignore success=ok] pam_localuser.so
auth sufficient pam_unix.so nullok try_first_pass
auth requisite  pam_succeed_if.so uid >= 1000
➥  quiet_success
auth sufficient pam_sss.so forward_pass
auth required   pam_deny.so

account required   pam_unix.so
account sufficient pam_localuser.so
account sufficient pam_succeed_if.so uid < 1000 quiet
account [default=bad success=ok user_unknown=ignore]
➥  pam_sss.so
account required   pam_permit.so

password requisite  pam_pwquality.so try_first_pass
➥  local_users_only
password sufficient pam_unix.so sha512 shadow nullok
➥  try_first_pass use_authtok
password sufficient pam_sss.so use_authtok
password required   pam_deny.so
```

Listing 21.6 *Von* `realmd` *modifizierte* `/etc/pam.d/system-auth` *leitet die Systemanmeldung an* `sssd` *weiter.*

[3] Bei anderen Unix-Systemen liegt die PAM-Konfiguration mitunter vollständig in der Datei `/etc/pam.conf`, unter Linux ist die Aufteilung auf einzelne Dateien im Verzeichnis `/etc/pam.d` üblich.

```
session  optional pam_keyinit.so revoke
session  required pam_limits.so
-session optional pam_systemd.so
session  optional pam_oddjob_mkhomedir.so umask=0077
session  [success=1 default=ignore] pam_succeed_if.so
➥  service in crond quiet use_uid
session  required pam_unix.so
session  optional pam_sss.so
```

An diesem Beispiel lassen sich die vier Aufgabenbereiche ablesen, für die PAM zuständig ist:

1. die *Authentisierung* im Sinne von Abschnitt 2.1 (Seite 11). In der PAM-Konfiguration als `auth` bezeichnet.
2. die *Zugriffskontrolle* im Sinne von Abschnitt 2.4 (Seite 22). Hier wird geprüft, ob ein Anwender für die Anmeldung *autorisiert* ist. In der PAM-Konfiguration als `account` bezeichnet.
3. die *Sitzungsverwaltung*. Hier können beispielsweise Umgebungsvariablen gesetzt oder das Heimatverzeichnis kann zur Verfügung gestellt werden. In der PAM-Konfiguration als `session` bezeichnet.
4. Schließlich unterstützt PAM die *Passwortänderung*. In der PAM-Konfiguration als `password` bezeichnet.

PAM-Stack

Pro Aufgabenbereich (`auth`, `account` `session` und `password`) können mehrere Module konfiguriert werden, die dann in einem Stapel abgearbeitet werden. Durch die folgenden *Schlüsselwörter* in der PAM-Konfiguration kann man pro PAM-Modul festlegen, wie sich sein Erfolg oder Misserfolg auf den gesamten Anmeldevorgang auswirken soll:

`required` Ist ein Modul als `required` konfiguriert, dann führt sein Misserfolg zum Fehlschlagen der Anmeldung, wobei aber noch die weiteren Module der PAM-Konfiguration abgearbeitet werden.

`requisite` Dies entspricht im Wesentlichen `required`; im Gegensatz zu diesem wird aber unmittelbar nach dem Modul abgebrochen, wenn dieses erfolglos war.

`sufficient` Der Erfolg eines mit `sufficient` konfigurierten Moduls genügt für eine erfolgreiche Anmeldung.

`optional` Das Modul wird ausgeführt, beeinflusst den Erfolg oder Misserfolg des Anmeldevorgangs aber nicht.

Komplexere Steuerungsmöglichkeiten existieren, sollen hier aber nicht weiter behandelt werden. Detaillierte Informationen liefert ein Blick in die Manual Page (`man 5 pam.conf`) zur PAM-Konfiguration.

Hinweis

Weder PAM- noch NSS-Konfiguration berücksichtigen, dass es sich hier um eine Active-Directory-Umgebung handelt. Sie sind sehr allgemein gehalten und daher für viele Anwendungsfälle bereits passend. Die Details der Anbindung verlagern sich auf den `sssd`. Der zentrale Dienst vereinfacht die Integration von Linux-Systemen selbst in komplexe Umgebungen ganz erheblich.

21.3.4 Erweiterte sssd-Konfiguration

Active Directory mit sssd

Die mit `realm join` automatisch erzeugte Konfiguration ist mit sinnvollen Standardwerten belegt und führt bereits zu einem funktionierenden System. Darüber hinaus lässt sich das Verhalten des `sssd` noch sehr detailliert weiter an die individuellen Anforderungen einer Umgebung anpassen. Beispielsweise vermeiden die mit Domänensuffix qualifizierten Benutzernamen zwar potenzielle Konflikte, unter Linux sind solche langen Namen jedoch unüblich. Sorgen etwa Namenskonventionen der Umgebung dafür, dass keine Namenskollisionen auftreten können, reicht auch ein kurzer Name ohne Suffix völlig aus. Der Parameter `use_fully_qualified_names` legt fest, ob `sssd` nur lange Benutzernamen samt Suffix akzeptiert. Falls Namenskollisionen nicht ausgeschlossen werden können, legt `domain_resolution_order` im allgemeinen Abschnitt `[sssd]` fest, welche Domänen bei der Suche Priorität erhalten. Listing 21.7 zeigt eine mögliche Konfiguration dafür.

```
[sssd]
domains = ads.example.com
config_file_version = 2
services = nss, pam
domain_resolution_order = ads.example.com, mydom.ads.
➥  example.com, otherdom.ads.example.com
full_name_format = %1$s

[domain/ads.example.com]
ad_domain = ads.example.com
krb5_realm = ADS.EXAMPLE.COM
realmd_tags = manages-system joined-with-adcli
cache_credentials = True
id_provider = ad
krb5_store_password_if_offline = True
default_shell = /bin/bash
ldap_id_mapping = True
```

Listing 21.7 *Modifizierte* `sssd`*-Konfiguration für kurze Benutzernamen*

```
use_fully_qualified_names = False
fallback_homedir = /home/%u
access_provider = ad
```

Wollen Sie kurze Benutzernamen nur aus der primären Domäne akzeptieren, muss der Parameter `use_fully_qualified_names` für die darunter liegenden Subdomänen angepasst werden. Die sind jedoch im Normalfall gar nicht explizit konfiguriert, sondern nur implizit über die Vertrauensstellung zur Hauptdomäne. In diesem Fall lassen sich Abweichungen über zusätzliche `domain`-Abschnitte erfassen, wie in Listing 21.8 gezeigt. Der Parameter `domains` muss dabei nach wie vor nur auf die Hauptdomäne und nicht auf den zusätzlichen Abschnitt verweisen.

Listing 21.8 *Abweichende Konfiguration für Subdomänen in* `sssd.conf`

```
[sssd]
domains = ads.example.com
[...]
[domain/ads.example.com/mydom.ads.example.com]
use_fully_qualified_names = True

[domain/ads.example.com/otherdom.ads.example.com]
use_fully_qualified_names = True
```

Optional greift `sssd` auch in die Kerberos-Konfiguration des Systems ein. Im Verzeichnis `/var/lib/sss/pubconf/krb5.include.d/` legt der Dienst darin Konfigurationsschnipsel ab, die sich bei Bedarf über `includedir`-Anweisungen in `/etc/krb5.conf` integrieren lassen. Ein konkretes Beispiel dafür lernen Sie in Abschnitt 22.2.2 kennen.

Eine weitere Anpassung der Standardvorgaben taucht so häufig auf, dass selbst das einfache Kommando `realm join` eine eigene Option dafür erhalten hat: die Abbildungsvorschrift für die numerischen Benutzer- und Gruppen-IDs nämlich. Wie in Abschnitt 21.3.1 beschrieben, berechnet `sssd` die IDs automatisch aus den Windows-SIDs der jeweiligen Objekte, wenn der Parameter `ldap_id_mapping` aktiviert ist. Diese Einstellung funktioniert ohne weitere Randbedingungen, gibt aber die Kontrolle über die IDs aus der Hand. Setzen innerhalb einer Umgebung beispielsweise die Firmenrichtlinien spezielle IDs voraus oder sollen im Rahmen einer Migration bestehende IDs erhalten bleiben, muss dieser Parameter auf `False` gesetzt oder `realm join` mit der Option `--automatic-id-mapping=no` aufgerufen werden. Der `sssd` erwartet dann, dass in Active Directory an jedem unter Linux verfügbaren Benutzer- und Gruppenobjekt explizite Attribute gepflegt werden, die die gewünschten numerischen IDs erhalten. Den dazu nötigen Schritten aus Sicht des Verzeichnisdienstes widmet sich der nachfolgende Abschnitt 21.4 im Detail.

FreeIPA mit sssd

Für integrierte Umgebungen auf Basis des in Kapitel 17 behandelten FreeIPA gelten aus Sicht des `sssd` im Wesentlichen dieselben Aussagen wie für Active Directory. In der Konfiguration ändert sich primär der benötigte Provider von `ad` zu `ipa`. Ein einfaches `realm join` erkennt den Typ der angegebenen Domäne sogar automatisch und spielt unmittelbar eine passende Konfiguration ein. Die Projekte FreeIPA und `sssd` sind eng miteinander verzahnt. In IPA-Umgebungen stehen deshalb noch einige weitere spezifische Funktionen zur Verfügung, wie beispielsweise die zentrale Verteilung von SELinux-Policies. Aus Sicht der grundlegenden Authentisierung und Benutzerverwaltung ähneln sich die Konfigurationsmöglichkeiten für Active Directory und IPA jedoch stark. Listing 21.9 zeigt einen Ausschnitt der `sssd.conf` eines IPA-Clientsystems.

Listing 21.9 *Von* `ipa-client-install` *erzeugte* `sssd.conf`

```
[sssd]
domains = ipa.example.com
services = nss, pam, ssh, sudo

[domain/ipa.example.com]
id_provider = ipa
ipa_server = _srv_, kdc01.ipa.example.com
ipa_domain = ipa.example.com
ipa_hostname = lx02.ipa.example.com
auth_provider = ipa
chpass_provider = ipa
access_provider = ipa
cache_credentials = True
ldap_tls_cacert = /etc/ipa/ca.crt
krb5_store_password_if_offline = True
[...]
```

Kerberos/LDAP-Umgebungen mit sssd

Mit reinem Kerberos und LDAP wie in den MIT- und Heimdal-Realms der Beispielumgebung ist mehr Handarbeit gefragt, denn die Automatismen und Konventionen der integrierten Lösungen stehen hier nicht zur Verfügung. Zwar lassen sich auch hier die oben beschriebenen Anpassungen für NSS und PAM einfach übernehmen. Das bislang verwendete, einfache Werkzeug `realm join` läuft jedoch ins Leere. Sie können stattdessen aber direkt auf das zugrunde liegende Kommando `authselect select sssd` zurückgreifen, optional ergänzt durch `with-mkhomedir`, um noch nicht vorhandene Heimatverzeichnisse automatisch beim ersten Login

anlegen zu lassen. Eine beispielhafte Konfiguration für `sssd` entnehmen Sie Listing 21.10.

Listing 21.10 *Konfiguration für reine Kerberos- und LDAP-Umgebungen in* `sssd.conf`

```
[sssd]
domains = mit.example.com, h5l.example.com
services = nss, pam
config_file_version = 2

[domain/mit.example.com]
id_provider = ldap
ldap_uri = ldap://kdc01.mit.example.com
ldap_search_base = dc=mit,dc=example,dc=com
ldap_schema = rfc2307
ldap_sasl_mech = GSSAPI
auth_provider = krb5
krb5_server = kdc01.mit.example.com
krb5_realm = MIT.EXAMPLE.COM
krb5_validate = true

[domain/h5l.example.com]
id_provider = ldap
ldap_uri = ldap://kdc01.h5l.example.com
ldap_search_base = dc=h5l,dc=example,dc=com
ldap_schema = rfc2307
ldap_sasl_mech = GSSAPI
auth_provider = krb5
krb5_server = kdc01.h5l.example.com
krb5_realm = h5l.EXAMPLE.COM
krb5_validate = true
```

Abgesehen von den DNS- und Realm-Namen unterscheiden sich die Einstellungen der MIT- nicht von der Heimdal-Umgebung. Im Gegensatz zu den zuvor betrachteten integrierten Lösungen müssen hier nun `id_provider` und `auth_provider` explizit und voneinander abweichend konfiguriert sein, denn Kerberos soll sich um die Authentisierung kümmern, während LDAP die restlichen Benutzerinformationen liefert. Die Angabe der KDCs in `krb5_server` kann entfallen, wenn passende Service-Records im DNS verfügbar sind. Der Name des Kerberos Realm in `krb5_realm` muss hingegen zwingend angegeben werden. Einen `subdom_provider`, der Abhängigkeiten zwischen Domänen automatisch erkennt, gibt es nur für die integrierten Umgebungen. Um Konfigurationsunterschiede zwischen vertrauenswürdigen Realms abzubilden, müssen hier stattdessen eigenständige Domänenabschnitte verwendet werden. Verlangt der LDAP-Server per SASL authentisierte Zugriffe, muss der ge-

wünschte Mechanismus in `ldap_sasl_mech` gesetzt werden. Die Provider `ipa` und `ad` stellen das automatisch ein.

Achtung

Die Sicherheit bei der Passwortüberprüfung wurde noch nicht diskutiert. Der Provider `krb5` des `sssd` holt sich mit den Anmeldeinformationen (Nutzername und Passwort) ein TGT. Lässt sich darin der Client-Teil mit dem Passwort entschlüsseln, dann kann es davon ausgehen, dass der Nutzer das Passwort eingegeben hat, das auch dem auf KDC-Seite gespeicherten Schlüssel entspricht. Zusätzlich muss aber auch die Authentizität des KDC überprüft werden. Ein Angreifer könnte sonst ein gefälschtes KDC verwenden, um sich Zugang zum System zu verschaffen. Um diesen sogenannten *Zanarotti-Angriff* unmöglich zu machen, muss der `krb5`-Provider das TGT validieren. Dies tut er, indem er mit dem TGT einen TGS-REQ für ein Host Ticket ausführt und dieses gegen die Keytab-Datei prüft. Ohne Host Keys in `/etc/krb5.keytab` oder mit der Standardeinstellung für `krb5_validate` auf `false` wäre dieses Angriffsszenario im Prinzip möglich. Die Provider für `ad` und `ipa` aktivieren die Validierung bereits automatisch.

Zanarotti-Angriff

Die LDAP-Parameter enthalten die URI und Suchbasis des zu befragenden Verzeichnisses, aber keine Angaben zur Authentisierung der Zugriffe. Das liegt am sinnvollen Standardverhalten des `sssd`, das eine explizite Konfiguration überflüssig macht: Ist in `/etc/krb5.keytab` ein Host Principal verfügbar, baut der Dienst damit automatisch SASL-authentisierte LDAP-Verbindungen auf. Falls Sie den Host-Principal bislang noch nicht per `kadmin` erstellt und in der Keytab der Linux-Clientmaschinen hinterlegt haben, sollten Sie dies nun nachholen. Mit den Parametern `ldap_sasl_authid` und `ldap_krb5_keytab` können Sie innerhalb des Konfigurationsabschnitts für jede Domäne bei Bedarf einen anderen Principal und eine andere Keytab vorgeben, die `sssd` zur Authentisierung in LDAP-Verbindungen verwenden soll. Bei der Angabe zur LDAP-Suchbasis können Sie optional auch Scope und Filter detailliert einstellen, beispielsweise mit `ldap_search_base = dc=mit,dc=example,dc=com?onelevel?(myattr=*)`. Noch mehr Kontrolle erlauben spezifische Parameter, die die Suchbasis für jeden einzelnen Objekttyp getrennt festlegen, etwa `ldap_user_search_base` für Benutzer und `ldap_group_search_base` für Gruppen.

Bleibt noch der Parameter `ldap_schema`. Er legt das genaue Format der erwarteten Objektklassen und Attribute in den LDAP-Abfragen fest, um alle benötigten Daten zu beziehen. Aktuell sind diese Informationen noch nicht in die Verzeichnisse eingespielt. Die folgenden Abschnitte werden das ändern. Auf Clientseite ist die Konfiguration vorerst abgeschlossen.

21.4 Ausbau der Gesamtstruktur

Die Gesamtstruktur dieses Buches besteht seit Kapitel 15 aus den zehn[4] verschiedenen Kerberos Realms, wie sie in Abbildung 6.5 auf Seite 113 dargestellt sind. Die Linux Realms sollten auch einen OpenLDAP-Dienst auf ihren KDC-Maschinen verwenden, um dort ihre Kerberos-Daten zu speichern. Falls das in Ihrer Umgebung noch nicht der Fall ist, so können Sie wie in Kapitel 13 und 14 vorgehen, um diesen Schritt nachzuholen.

Alle LDAP-Dienste kerberisieren

Danach sollten Sie die Kerberisierung aller OpenLDAP-Dienste analog zu Kapitel 20 durchführen. Für die Active Directory Realms entfallen diese Schritte selbstverständlich.

Ein LDAP-Dienst pro Kerberos Realm

Im Folgenden wird davon ausgegangen, dass es in der Gesamtstruktur kerberisierte LDAP-Server für die folgenden Verzeichnisbäume gibt:

- `dc=example,dc=com`
- `dc=mit,dc=example,dc=com`
- `dc=mydomain,dc=mit,dc=example,dc=com`
- `dc=otherdomain,dc=mit,dc=example,dc=com`
- `dc=h5l,dc=example,dc=com`
- `dc=mydomain,dc=h5l,dc=example,dc=com`
- `dc=otherdomain,dc=h5l,dc=example,dc=com`
- `dc=ads,dc=example,dc=com`
- `dc=mydomain,dc=ads,dc=example,dc=com`
- `dc=otherdomain,dc=ads,dc=example,dc=com`

21.4.1 LDAP-Referrals einrichten

Dieser Schritt ist optional und wird hier nur der Vollständigkeit halber behandelt. Denn die Funktionalität der LDAP-Referrals wird im weiteren Verlauf dieses Buches nicht weiter genutzt werden.

Grundsätzlich benutzt ein LDAP-Server Referrals dazu, einem anfragenden LDAP-Client mitzuteilen, dass er keine oder nur unvollständige Informationen liefern kann und dass der Client noch weitere LDAP-Server kontaktieren muss, um eine umfassende Antwort zu erhalten. Beispielsweise kann der LDAP-Server, der für `dc=example,dc=com` zuständig ist, keine Informationen über Objekte unterhalb von `dc=mit,dc=example,dc=com` liefern, da diese unter der Hoheit eines anderen LDAP-Servers stehen. Diese Art von Referrals nennt man auch *Subordinate Referrals*, da sie in der Hierarchie nach unten verweisen. Sie sind sozusagen Delegationspunkte,

Subordinate Referrals

[4] Statt Active Directory können Sie analog auch die Samba- oder IPA-Installationen verwenden. Für die weitere Betrachtung sind die Unterschiede zwischen den Implementierungen unwesentlich.

um einen Teilbaum einem anderen Server zuzuordnen. Für die andere Richtung sind *Superior Referrals* zuständig, die beispielsweise dann zum Tragen kämen, wenn man bei dem für `dc=mit,dc=example,dc=com` zuständigen LDAP-Servern nach Objekten in `dc=example,dc=com` suchen würde.

Superior Referrals

Subordinate Referrals werden durch LDAP-Objekte erzeugt, wie sie Listing 21.11 darstellt. Das Referral-Objekt ist von der Objektklasse `extensibleObject`, damit kann es beliebige Attribute annehmen. Der Wert des Attributs `ref` ist der Verweis, der an den Client zurückgeschickt werden soll.

Listing 21.11
Ein Referral-Objekt

```
dn: dc=mit,dc=example,dc=com
objectClass: referral
objectClass: extensibleObject
dc: mit
ref: ldap://kdc01.mit.example.com/dc=mit,dc=example,dc=com
```

Superior Referrals werden nicht durch Objekte im Baum repräsentiert, sondern sind Teil der Serverkonfiguration. Bei OpenLDAP erfolgt diese Konfiguration über das Attribut `olcReferral` in `cn=config`. Beispielsweise muss dazu auf den für `dc=mit,dc=example,dc=com` zuständigen LDAP-Servern `olcReferral` den Wert `ldap://kdc01.example.com` erhalten.

Wenn Sie für alle LDAP-Teilbäume hier entsprechende Objekte und Konfigurationen erzeugen, dann erhalten die LDAP-Clients immer korrekte Referrals. Für den Active Directory Forest unterhalb von `dc=ads,dc=example,dc=com` ist dabei übrigens nichts zu tun: AD konfiguriert die LDAP-Referrals automatisch beim Erzeugen der Subdomänen.

Das Problem mit den Referrals sind aber die LDAP-Clients, die den Referrals nachgehen und dazu eine entsprechende Logik implementieren müssen. Manche Clients unterstützen Referrals aber gar nicht. Die meisten Clients, die Referrals nachgehen können, tun dies nur ohne Authentisierung. Daher sollte man in der Praxis nicht zu sehr auf diese LDAP-Funktionalität setzen.

21.4.2 Identitäts- und Autorisierungsdaten für Linux

Hier geht es nun darum, die verschiedenen LDAP-Bäume der beschriebenen Gesamtinfrastruktur mit Benutzerdaten zu füllen. Bisher existieren ja nur vereinzelte Einträge für Max Mustermann und Erika Musterfrau oder einige wenige administrative LDAP-Accounts. Nun sollen weitere Objekte folgen.

Die bisherigen LDAP-Nutzereinträge, wie Sie sie für OpenLDAP in Listing 13.10 auf Seite 262 oder für Active Directory in Listing 15.18 auf Seite 341 gesehen haben, reichen für den Zweck der Linux-Anbindung

nicht aus. Denn die beiden OpenLDAP-Objekte besitzen keine Nutzerdaten, wie man sie für Linux-Accounts benötigt, und die AD-Objekte haben lediglich Windows-Attribute. Für die Nutzung unter Linux werden also Nutzerobjekte mit weiteren Attributen benötigt, und auch Gruppenobjekte müssen angelegt werden.

POSIX-Daten

Die erforderlichen Linux-Nutzerdaten werden auch *POSIX-Daten* genannt. Das sind genau die Daten, die traditionell in den beiden Dateien `/etc/passwd`[5] und `/etc/group` (siehe auch Listings 21.12 und 21.13) zu finden sind. Unter *Unix-Nutzer-* oder *»passwd«-Daten* versteht man die folgenden Felder aus der letzten Zeile in Listing 21.12:

Listing 21.12 *Nutzerdaten in* `/etc/passwd` *(gekürzt). Die letzte Zeile stellt die allgemeine Form eines Nutzereintrags dar.*

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
man:x:6:12:man:/var/cache/man:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
testuser1:x:998:998:Testnutzer Nr. 1:/home/testuser1:/bin/ ↩
➥  bash
testuser2:x:999:999:Testnutzer Nr. 2:/home/testuser1:/bin/ ↩
➥  bash
[...]
Benutzername:PW-Hash:UID:GID:Gecos:Heimatverzeichnis:Shell
```

Listing 21.13 *Gruppendaten in* `/etc/group` *(gekürzt). Die letzte Zeile stellt die allgemeine Form eines Gruppeneintrags dar.*

```
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:testuser1,testuser2
tty:x:5:
disk:x:6:
[...]
Gruppenname:PW-Hash:GID:Mitgliederliste
```

Benutzername Der Unix-Account-Name des Anwenders

PW-Hash Ein nicht klartextäquivalenter Hash des Anwenderpasswortes (auch als *»Crypt-String«* bezeichnet, siehe auch Abschnitt 4.2.3)

[5] Der *PW-Hash* befindet sich typischerweise in der separaten Datei `/etc/shadow`.

UID Die numerische Nutzer-ID des Anwenders
GID Die numerische ID seiner *primären* Gruppe
Gecos Eine kurze Beschreibung des Accounts. Häufig werden hier Vor- und Nachname oder auch Telefonnummern und E-Mail-Adressen eingetragen.
Heimatverzeichnis Das Heimatverzeichnis des Anwenders
Shell Die Login-Shell des Anwenders, also das Programm, das direkt nach dem Login gestartet wird

Entsprechend versteht man unter *Gruppendaten* die folgenden Felder aus Listing 21.13:

Gruppenname Der Name der Gruppe
PW-Hash Ein nicht klartextäquivalenter Passwort-Hash der Gruppe (auch als »*Crypt-String*« bezeichnet). Dieses Feld wird heutzutage kaum verwendet und durch ein »x« gesperrt.
GID Die numerische ID der Gruppe
Mitgliederliste Die Liste der Nutzer-Accounts, die Mitglied in dieser Gruppe sind, durch Kommata getrennt. Diese Liste kann auch leer sein. Daraus ergeben sich die sogenannten *sekundären Gruppen* eines Nutzers.

RFC 2307

All diese POSIX-Daten muss ein LDAP-Server anbieten können, falls er als Verzeichnisdienst für Unix- oder Linux eingesetzt werden soll. Es stellt sich dabei die Frage, in welcher Form, sprich mit welchen Objektklassen und in welchen Attributen diese Daten im LDAP zu speichern sind. Beantwortet wird diese Frage durch einen Internetstandard aus dem Jahre 1998: RFC 2307 [10][6]. Dieser definiert Attribute und Objektklassen für allgemeine Netzwerkinformationen für Unix-artige Systeme, darunter auch für Nutzer- und Gruppeninformationen. Für Nutzerdaten sieht RFC 2307 die Objektklasse `posixAccount` und folgende Attribute vor:

RFC 2307 definiert Objektklassen und Attribute für POSIX-Daten im LDAP.

uid für *Benutzername*
userPassword für den *PW-Hash*
uidNumber für die *UID*
gidNumber für die *GID*
gecos für den *Gecos*
homeDirectory für das *Heimatverzeichnis*

Attribute für `posixAccount`

[6] Eine etwas neuere Version dieses Standards ist *RFC 2307bis*, ein Internet-Draft aus dem Jahre 2002, der allerdings im April 2003 abgelaufen ist und nie den vollen RFC-Status erlangt hat [11]. RFC 2307bis bringt einige Vorteile mit sich, beispielsweise ist `posixGroup` nicht mehr structural.

loginShell für die *Shell*

Für Gruppendaten sieht RFC 2307 die Objektklasse posixGroup und folgende Attribute vor:

Attribute für posixGroup

cn für *Gruppenname*
userPassword für den *PW-Hash* (optional, wird sehr selten verwendet)
gidNumber für die *GID*
memberUid für die *Mitgliederliste*[7]

POSIX-Objekte für OpenLDAP

OpenLDAP-Schemaerweiterung bereits erledigt

Die zugehörigen Schemadefinitionen liegen auf den CentOS-Systemen in /etc/openldap/schema/nis.ldif und wurden für den Server kdc01.example.com bereits in Kapitel 7 eingespielt (Listing 7.15 auf Seite 152). Eine Schemaerweiterung ist hier also nicht mehr nötig, und auch die anderen OpenLDAP-Dienste der Beispielumgebung dieses Buches sollten für die Aufnahme von POSIX-Attributen bereits vorbereitet sein. Falls nicht, hilft folgendes Kommando: ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/nis.ldif.

Listing 21.14 zeigt Beispielobjekte in LDIF-Syntax, die einen Linux-Account und eine Linux-Gruppe repräsentieren. Im Account-Objekt für Max Mustermann sind keine Kerberos-Eigenschaften enthalten. Diese sollten wie in Listing 13.12 auf Seite 263 mit kadmin gesetzt werden. Auch wird bewusst kein userPassword-Attribut gesetzt, da die Passwortüberprüfung über Kerberos erfolgen soll. Das zweite Beispiel aus Listing 21.14 zeigt eine Linux-Gruppe, die unter anderem maxm und erim als Mitglieder enthält. Beide Einträge sind nur als Beispiele gedacht.

Listing 21.14 *Beispielnutzer und Beispielgruppe für OpenLDAP. Die zusätzlichen Kerberos-Attribute setzt man am einfachsten mit* kadmin.

```
dn: cn=Max Mustermann,ou=people,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
cn: Max Mustermann
sn: Mustermann
uid: maxm
uidNumber: 10000
gidNumber: 123
gecos: Herr Mustermann
homeDirectory: /home/maxm
```

[7] RFC 2307bis ermöglicht hier auch die Verwendung der member- oder uniqueMember-Attribute, die auch für andere Gruppenobjekte im LDAP verwendet werden.

```
loginShell: /bin/bash

dn: cn=Musterleute,ou=groups,dc=example,dc=com
objectClass: posixGroup
cn: Musterleute
gidNumber: 123
memberUid: maxm
memberUid: erim
```

POSIX-Objekte für Active Directory

Auch Active Directory hat seit Windows Server 2003 R2 und auch in der hier verwendeten Version 2019 ein LDAP-Schema (das »R2-Schema«), das zu RFC 2307 weitgehend kompatibel ist. Eine Schemaerweiterung für die POSIX-Eigenschaften ist bei aktuellen Windows-Versionen also ebenfalls unnötig. In ein paar Dingen unterscheiden sich die Schemata aber schon:

Keine Schemaerweiterung bei aktueller Version

- Der Name des Attributs für das Heimatverzeichnis lautet im Active Directory abweichend vom RFC 2307 `unixHomeDirectory`.
- Der Name der Objektklasse für Anwenderobjekte lautet schlicht `User`, der für Gruppen `Group`.
- Unix-Gruppenzugehörigkeiten können über das RFC 2307-Attribut `memberUid` gesetzt werden, native Windows-Gruppenmitgliedschaften werden aber im Attribut `member` gespeichert, dort aber nicht in Form eines kurzen Account-Namens, sondern in Form des DN des Mitgliedseintrags. Das entspricht dem RFC 2307bis (siehe Fußnote 6 auf Seite 469), der bei der im weiteren Verlauf dieses Kapitels beschriebenen Linux-Anbindung allerdings nicht verwendet werden soll.

Listing 21.15 stellt entsprechende Beispielobjekte für Active Directory dar. Das Account-Objekt entspricht dem aus Listing 15.18 von Seite 341 und ist nur um die POSIX-Eigenschaften erweitert. Auch hier sind beide Einträge nur als Beispiele gedacht.

```
dn: CN=Erika Musterfrau,CN=Users,DC=ADS,DC=EXAMPLE,DC=COM
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: Erika Musterfrau
sn: Musterfrau
givenName: Erika
```

Listing 21.15 *Beispielnutzer und Beispielgruppe für Active Directory*

```
instanceType: 4
displayName: Erika Musterfrau
name: Erika Musterfrau
userAccountControl: 512
sAMAccountName: erim
userPrincipalName: erim@ADS.EXAMPLE.COM
unicodePwd:: IgBTAHQAYQByAHQAMQAyADMAIgA=
pwdLastSet: 0
uid: erim
uidNumber: 10001
gidNumber: 123
gecos: Frau Musterfrau
unixHomeDirectory: /home/erim
loginShell: /bin/bash

dn: CN=Musterleute,CN=Users,DC=ADS,DC=EXAMPLE,DC=COM
objectClass: top
objectClass: group
cn: Musterleute
sAMAccountName: Musterleute
gidNumber: 123
memberUid: maxm
memberUid: erim
```

Hinweis

Erstrecken sich die Benutzerobjekte in einem Active Directory Forest über mehrere Domänen, sollten die Posix-Attribute aus Performancegründen in den Global Catalog repliziert werden. Andernfalls muss `sssd` für die benötigten Daten jede Domäne einzeln nacheinander abfragen.

Benutzer- und Gruppenobjekte anlegen

Hier sollen nun Benutzer- und Gruppenobjekte angelegt werden, die im weiteren Verlauf dieses Buches verwendet werden sollen. Die folgenden Kapitel gehen davon aus, dass die LDAP-Verzeichnisse der verschiedenen Beispiel-Realms mit Nutzer- und Gruppendaten aus den Bereichen gefüllt sind, wie sie Tabelle 21.1 auf Seite 473 zeigt. Wenn Sie diese Benutzer- und Gruppenobjekte in Ihrem Testnetz ebenfalls anlegen wollen, dann können Sie die LDIF-Beispiele aus den Listings 21.14 und 21.15 als Vorlage verwenden. Mithilfe der Linux-Shell lässt sich der Vorgang leicht automatisieren.

Realm	Benutzerbereich Namen / UID	Gruppenbereich Namen / GID
MIT.EXAMPLE.COM	user1001 – user1999 1001 – 1999	group1001 – group1999 1001 – 1999
MYDOM.MIT.EXAMPLE.COM	user2001 – user2999 2001 – 2999	group2001 – group2999 2001 – 2999
OTHERDOM.MIT.EXAMPLE.COM	user3001 – user3999 3001 – 3999	group3001 – group3999 3001 – 3999
H5L.EXAMPLE.COM	user4001 – user4999 4001 – 4999	group4001 – group4999 4001 – 4999
MYDOM.H5L.EXAMPLE.COM	user5001 – user5999 5001 – 5999	group5001 – group5999 5001 – 5999
OTHERDOM.H5L.EXAMPLE.COM	user6001 – user6999 6001 – 6999	group6001 – group6999 6001 – 6999
ADS.EXAMPLE.COM	user7001 – user7999 7001 – 7999	group7001 – group7999 7001 – 7999
MYDOM.ADS.EXAMPLE.COM	user8001 – user8999 8001 – 8999	group8001 – group8999 8001 – 8999
OTHERDOM.ADS.EXAMPLE.COM	user9001 – user9999 9001 – 9999	group9001 – group9999 9001 – 9999

Tab. 21.1
Benutzer- und Gruppenbereiche für die Beispiel-Realms

21.5 Linux-Clients in der Gesamtinfrastruktur

21.5.1 Anbindung testen

Nachdem Sie Ihre Linux-Clients in Abschnitt 21.3 so konfiguriert haben, dass Benutzerinformationen über den zentralen Verzeichnisdienst der jeweiligen Umgebung abgerufen werden und in Abschnitt 21.4.2 die Verzeichnisse mit passenden Daten worden sind, geht es nun ans Testen der Integration. Listing 21.16 zeigt das beispielhaft anhand des Clients `lx01.ads` in der Active-Directory-Umgebung. Entsprechende Ergebnisse sollten Sie auch auf den Clients der anderen Umgebungen erhalten. Die gezeigten Schritte überprüfen, dass Benutzernamen, Gruppennamen und die zugehörigen IDs aus dem Verzeichnis gelesen und passend zugeordnet werden.

Listing 21.16 *Einige Tests der Namensauflösung über Active Directory*

```
root@lx01.ads:~# touch /tmp/testfile
root@lx01.ads:~# chown user7495:group7001 /tmp/testfile
root@lx01.ads:~# ls -l /tmp/testfile
-rw-r--r--. 1 user7495 group7001 0 Sep 18 18:15 /tmp/
➥  testfile
root@lx01.ads:~# id user7495
uid=7495(user7495) gid=7495(group7495) groups=7495(
➥  group7495)
root@lx01.ads:~# getent passwd user7495
user7495:*:7495:7495:ADS Testuser 495:/home/user7495:/bin/
➥  bash
root@lx01.ads:~# su - user7495
Creating home directory for user7495.
[user7495@lx01.ads ~]$ whoami
user7495
```

Die `sssd`-Parameter `ldap_id_mapping` und `use_fully_qualified_names` sind im gezeigten Beispiel deaktiviert, um kurze Namen ohne Suffix und fest eingestellte numerische IDs für Benutzer und Gruppen zu erhalten. Experimentieren Sie in der Active-Directory-Umgebung auch mit anderen Einstellungen, aber achten Sie beim Wechsel darauf, den `sssd`-Cache zu verwerfen (`sss_cache -E`). Andernfalls arbeiten Sie möglicherweise mit veralteten Daten. Listing 21.17 zeigt die Auswirkungen von `ldap_id_mapping = True`: In der Beispielumgebung ändern sich dadurch nicht nur die numerischen IDs von Benutzern und Gruppen, sondern unter Umständen ändert sich auch die Gruppenzugehörigkeit selbst. Die primäre Gruppe `group7495` ist die numerische ID über das Posix-Attribut `gidNumber` zugeordnet. Automatisches ID-Mapping wertet dieses Attribut nicht mehr aus, und die Gruppenmitgliedschaft verschwindet. Umgekehrt gehört `user7495` nun neu zur Gruppe `domain users`, einer Windows-Standardgruppe, die ohne zugewiesene Posix-Attribute zuvor unter Linux gar nicht sichtbar war.

Listing 21.17 *Gruppenzuordnung mit automatischem ID-Mapping in Active Directory*

```
root@lx01.ads:~# id user7495
uid=518602595(user7495) gid=518600513(domain users) groups
➥  =518600513(domain users)
```

Versuchen Sie auch, sich mit einem Domänenbenutzer lokal über die Textkonsole, den grafischen Login-Bildschirm oder per `ssh` anzumelden. Das muss mit dem zentral hinterlegten Passwort klappen. Im Anschluss sollte der Benutzer auch bereits über einen befüllten Credential Cache mit gültigem TGT verfügen (siehe Listing 21.18).

```
root@lx01.ads:~# ssh -l user7495 lx01.ads.example.com
user7495@lx01.ads's password: P@ssw0rd
[user7495@lx01.ads ~]$ klist
Ticket cache: FILE:/tmp/krb5cc_7495_ DfK3feZYlZ
Default principal: user7495@ADS.EXAMPLE.COM

Valid starting     Expires            Service principal
09/18/21 18:58:18  09/19/21 04:58:18  krbtgt/ADS.EXAMPLE.
➥  COM@ADS.EXAMPLE.COM
        renew until 09/25/21 18:58:18
[user7495@lx01.ads ~]$
```

Listing 21.18
Nach der Anmeldung per Passwort steht bereits ein TGT zur Verfügung.

21.5.2 Multi-Domänen-Anbindung konfigurieren und testen

Die bisherigen Tests haben sich auf einfache Konfigurationen innerhalb einer Realm oder Domäne beschränkt. Wie in Abschnitt 21.3 bereits angeklungen ist, kann `sssd` aber mehr als das. Über den Dienst lassen sich die Benutzer aus mehreren Domänen ins System integrieren. Für Active Directory Forests passiert das sogar bereits automatisch. In den reinen Kerberos- und LDAP-Umgebungen müssen die einzelnen Domänen noch über separate Einträge in die Konfiguration des `sssd` aufgenommen werden.

Listing 21.19 zeigt eine minimale Konfigurationsdatei `sssd.conf`, um einen Linux-Client gleichzeitig sowohl an einen MIT Kerberos Realm mit zugehörigem LDAP-Server als auch einen Active Directory Forest anzubinden. Die Einstellung für `full_name_format` mit dem speziellen Wert `%1$s` sorgt für einheitlich kurze Namen ohne Domänensuffix über alle Domänen hinweg. Die Parameter `min_id` und `max_id` dienen der korrekten Domänenzuordnung bei der Rückwärtsauflösung von numerischen IDs zu Namen.

```
[sssd]
domains = mit.example.com, ads.example.com
services = nss, pam
config_file_version = 2
domain_resolution_order = ads.example.com, mydom.ads.
➥  example.com, otherdom.ads.example.com
full_name_format = %1$s

[domain/mit.example.com]
id_provider = ldap
ldap_uri = ldap://kdc01.mit.example.com
```

Listing 21.19
Beispielkonfiguration für gemischte Gesamtumgebung mit MIT-Realm und Active Directory

```
ldap_search_base = dc=mit,dc=example,dc=com
ldap_schema = rfc2307
ldap_sasl_mech = GSSAPI
ldap_sasl_authid = lx02$@ADS.EXAMPLE.COM
auth_provider = krb5
chpass_provider = krb5
access_provider = krb5
krb5_server = kdc01.mit.example.com
krb5_realm = MIT.EXAMPLE.COM
krb5_validate = true
min_id = 1001
max_id = 1999

[domain/ads.example.com]
id_provider = ad
ldap_id_mapping = False
min_id = 7001
max_id = 9999
```

Die gezeigte Konfiguration passt für Clients, deren Host-Principal der MIT-Realm oder der Wurzeldomäne des Active Directory Forests angehört. Ein an `mydom.ads.example.com` angebundener Client verwendet stattdessen eine abgewandelte `domain_resolution_order`, die die eigene Domäne an erster Stelle führt. In der `domains`-Liste und dem `domain`-Abschnitt wird `ads.example.com` jeweils durch `mydom.ads.example.com` ersetzt. Für Heimdal- oder weitere MIT-Realms fügen Sie analog zu `example.com` weitere `domain`-Einträge hinzu. Erweitern Sie die `sssd`-Konfigurationen der `lx01`- und`lx02`-Systeme der von Ihnen eingerichteten Testumgebungen auf Basis von Listing 21.19 so, dass diese sämtliche Realms umfassen.

Ist alles korrekt eingerichtet, können Sie die in Abschnitt 21.5.1 beschriebenen Tests für Benutzer aus den zusätzlichen verbundenen Domänen wiederholen.

21.5.3 Schattenobjekte für LDAP-Zugriff auf Active Directory

Die Linux-Systeme authentisieren sich gegenüber den LDAP-Diensten der Beispielumgebung mit ihren eigenen Host-Principals. So versucht beispielsweise der `sssd` der `lx01.mit.example.com`, sich als `host/lx01.mit.example.com@MIT.EXAMPLE.COM` zu authentisieren. Die `olcAccess`-Konfigurationen der OpenLDAP-Dienste sind entsprechend vorbereitet und gestatten prinzipiell jedem authentisierten Zugriff Leserechte auf alle Posix-Attribute, auch Principals aus fremden, per Trust ver-

bundenen Realms. Strikter verhalten sich die LDAP-Dienste der Active-Directory-Domänen. Sie erwarten zur Autorisierung im LDAP-Service-Ticket ein Windows-typisches Zugriffstoken. Für einen Principal wie `host/lx01.mit.example.com@MIT.EXAMPLE.COM` kennt Active Directory jedoch keine Autorisierungsdaten. Folglich fehlt auch das benötigte PAC-Feld im Service-Ticket.

Mit einem einfachen Trick kann man den KDCs der Active-Directory-Umgebung die externen Host-Principals aber dennoch bekannt machen. Dazu legt man als eine Art »Schattenobjekt« ein Computerkonto an und verknüpft es mit den Kerberos-Principal-Namen der anderen Realms. Listing 21.20 stellt ein solches Schattenobjekt in LDIF-Form dar[8].

Listing 21.20
Ein Schattenobjekt im AD für die Linux-Maschine `lx01`

```
dn: CN=nonad,CN=Computers,DC=ads,DC=example,DC=com
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
objectClass: computer
cn: nonad
userAccountControl: 4128
sAMAccountName: nonad$
unicodePwd:: IgBhAGUAUQB2AEcAagBpAEoAegBwAFYANgBoADMAQgBhA
 EIATQBFAFIANwBWAHEAbABDADUAeABTAEIAcQB2AEIAdwB4AEUAMQAzAE
 gAaABEACIA
altSecurityIdentities:
 Kerberos:host/lx01.mit.example.com@MIT.EXAMPLE.COM
altSecurityIdentities:
 Kerberos:host/lx02.mit.example.com@MIT.EXAMPLE.COM
altSecurityIdentities:
 Kerberos:host/lx01.mydom.mit.example.com@MYDOM.MIT.EXAMPL
 E.COM
altSecurityIdentities:
 Kerberos:host/lx02.mydom.mit.example.com@MYDOM.MIT.EXAMPL
 E.COM
altSecurityIdentities:
 Kerberos:host/lx01.otherdom.mit.example.com@OTHERDOM.MIT.
 EXAMPLE.COM
altSecurityIdentities:
 Kerberos:host/lx02.otherdom.mit.example.com@OTHERDOM.MIT.
 EXAMPLE.COM
```

[8] Das Attribut `unicodePwd` enthält aus Sicherheitsgründen ein zufällig erzeugtes Passwort.

```
altSecurityIdentities:
 Kerberos:host/lx01.h5l.example.com@H5L.EXAMPLE.COM
altSecurityIdentities:
 Kerberos:host/lx02.h5l.example.com@H5L.EXAMPLE.COM
altSecurityIdentities:
 Kerberos:host/lx01.mydom.h5l.example.com@MYDOM.H5L.EXAMPL
 E.COM
altSecurityIdentities:
 Kerberos:host/lx02.mydom.h5l.example.com@MYDOM.H5L.EXAMPL
 E.COM
altSecurityIdentities:
 Kerberos:host/lx01.otherdom.h5l.example.com@OTHERDOM.H5L.
 EXAMPLE.COM
altSecurityIdentities:
 Kerberos:host/lx02.otherdom.h5l.example.com@OTHERDOM.H5L.
 EXAMPLE.COM
```

Ist das Objekt `nonad$` aus Listing 21.20 eingespielt, erhalten alle Linux-Systeme der Beispielumgebung für LDAP-Abfragen des Active Directory die nötigen PAC-Daten. Damit sind nun Benutzer aus dem Active Directory Forest auch auf den Systemen der MIT- und Heimdal-Realms bekannt.

21.6 Zusammenfassung

In diesem Kapitel haben Sie die Anbindung von Betriebssystemen an zentrale Kerberos- und LDAP-Infrastrukturen kennengelernt. Die Anbindung von Windows-Clients beschränkte sich dabei auf die Active-Directory-Teilstruktur und war durch den Domain Join sehr einfach zu realisieren. Für die Anbindung der Linux-Maschinen sind die folgenden Linux-Subsysteme wesentlich: NSS für die Anbindung der Account-Informationen an LDAP-Verzeichnisse und PAM für die Anbindung der Authentisierung an Kerberos-Infrastrukturen. Beides wird heute über den `sssd` als zentralem Dienst abgewickelt, der auch mit komplexen Umgebungen aus mehreren Realms und Domänen umgehen kann.

Kerberos spielte in diesem Kapitel eine eher untergeordnete Rolle, es wurde lediglich als Dienst für die Passwortüberprüfung eingesetzt und lieferte Integrität und Vertraulichkeit für die LDAP-Daten durch SASL/GSS-API.

Mit der Anbindung der NSS an die LDAP-Infrastruktur in diesem Kapitel dieses Buches ist aber eine wichtige Voraussetzung für die folgenden Kapitel geschaffen, die sich dem Single Sign-on durch Kerberos

widmen werden und das Vorhandensein der Betriebssystem-Accounts voraussetzen.