stimmt, die von Kerberos noch akzeptiert wird (*Clock Skew*, siehe Abschnitt 5.1.1). Der Default ist fünf Minuten.

Max. Zeitraum, in dem ein Benutzerticket erneuert werden kann
Damit definiert man die Renewable Lifetime. Der Default-Wert beträgt sieben Tage.

Wenn Sie Änderungen an den beschriebenen Richtlinien machen, dann können Sie für diese durch das Kommando `gpupdate /force` eine sofortige Aktualisierung erwirken.

15.5 Keytab-Verwaltung in AD-Infrastrukturen

Kerberisierte Dienste unter Unix und Linux benötigen bekanntlich Keytab-Dateien. Unter Windows-Systemen sind diese in der Regel nicht nötig. Für die Erzeugung von Keytabs in AD-Umgebungen stehen verschiedene Werkzeuge zur Verfügung. Eines dieser Werkzeuge ist das Kommando `ktpass.exe`, das Bestandteil des Windows-Betriebssystems ist und das man daher auch unter Windows ausführen muss. Linux-Alternativen wie `msktutil` oder `adcli` sind einerseits komfortabler, in der Praxis muss man sich aber häufig auf die Verwaltungsmöglichkeiten von Windows beschränken. Daher soll `ktpass.exe` hier an erster Stelle behandelt werden.

15.5.1 Keytabs unter Windows mit ktpass.exe erzeugen

Das Werkzeug `ktpass.exe` kann Keytabs sowohl für Maschinenkonten als auch für die auf Seite 326 beschriebenen Dienste-Accounts erzeugen. Im folgenden Beispiel soll eine Keytab für einen Host Principal erzeugt werden.

```
C:\Users\Administrator>ktpass.exe /out lx01.keytab          ^
  /mapuser LX01$@ADS.EXAMPLE.COM                            ^
  /princ host/lx01.ads.example.com@ADS.EXAMPLE.COM          ^
  /rndPass                                                  ^
  /crypto AES256-SHA1                                       ^
  /ptype KRB5_NT_SRV_HST
Targeting domain controller: kdc01.ads.example.com
Using legacy password setting method
Successfully mapped host/lx01.ads.example.com to LX01$.
WARNING: Account LX01$ is not a user account (uacflags=0    ↵
➥  x1021).
```

Listing 15.8 *Unter Windows eine Keytab-Datei für den Host Principal erzeugen*

```
WARNING: Resetting LX01$'s password may cause                    ➥
➥  authentication problems if LX01$ is being used as a          ➥
➥  server.

Reset LX01$'s password [y/n]?  y
Key created.
Output keytab to lx01.keytab:
Keytab version: 0x502
keysize 92 host/lx01.ads.example.com@ADS.EXAMPLE.COM ptype       ➥
➥   3 (KRB5_NT_SRV_HST) vno 3 etype 0x12 (AES256-SHA1)          ➥
➥  keylength 32 (0x01839277d939fc874c3d96a882371e9               ➥
➥  90536f4637823a83415c16834fc14e8a6)

C:\Users\Administrator>
```

Listing 15.8 zeigt, wie Sie unter Windows mit `ktpass.exe` eine Keytab-Datei für den Host Principal der `lx01` schreiben können. Dabei wird das Maschinenpasswort, das im AD-Objekt `LX01$` gespeichert ist, auf einen zufälligen Wert gesetzt (dafür sorgt die Option `/rndPass`) und ein AES-256-Schlüssel erzeugt, der dann in die Datei `lx01.keytab` exportiert wird.[6] Nebenbei werden auch SPN und UPN gesetzt, sodass die für Kerberos wesentlichen Namensattribute am Objekt der `LX01$` anschließend folgendermaßen aussehen:

- `userPrincipalName: host/lx01.ads.example.com@ADS.EXAMPLE.COM`
- `servicePrincipalName: host/lx01.ads.example.com` und `HOST/LX01`
- `sAMAccountName: LX01$`

Bringen Sie diese Keytab auf einem sicheren Weg auf die Maschine `lx01` und speichern Sie sie dort unter `/etc/krb5.keytab` ab. Achten Sie dabei auch auf die Zugriffsrechte: `chmod 600 /etc/krb5.keytab`.

Listing 15.9 zeigt, wie Sie die Keytab auf der `lx01` anzeigen und überprüfen können. Das `kinit`-Kommando liefert nun keine Fehlermeldung mehr, da jetzt auch der `userPrincipalName` gesetzt ist. `kvno` testet zum einen die Verfügbarkeit des Service-Principal im AD, durch die Option `-k /etc/krb5.keytab` werden die enthaltenen Schlüssel auch gleich auf Korrektheit geprüft.

Listing 15.9 *Test der Keytab unter Linux*

```
root@lx01.ads:~# klist -ke
Keytab name: FILE:/etc/krb5.keytab
```

[6] Das Kommando schreibt auch den Key in hexadezimales Form (`x0183927…`) auf die Standardausgabe. Häufig werden solche Ausgaben dann in eine Dokumentation kopiert (wie auch hier das Listing 15.8). In Produktionsumgebungen gilt es dies tunlichst zu vermeiden.

```
KVNO Principal
---- --------------------------------------------------
   4 host/lx01.ads.example.com@ADS.EXAMPLE.COM (aes256-cts ➥
➥  -hmac-sha1-96)
root@lx01.ads:~# kinit -k host/lx01.ads.example.com
root@lx01.ads:~# kvno -k /etc/krb5.keytab                  \
                       host/lx01.ads.example.com
host/lx01.ads.example.com@ADS.EXAMPLE.COM: kvno = 4,       ➥
➥  keytab entry valid
root@lx01.ads:~#
```

15.5.2 Host Keytabs unter Linux mit adcli verwalten

CentOS-Linux enthält das Kommando `adcli`, das Sie über per `dnf install adcli` installieren können. Im Gegensatz zu `ktpass.exe` arbeitet man mit `adcli` direkt auf der Linux-Kommandozeile. Mit diesem Kommando und der Option `join` können Sie gewissermaßen einen »Domain Join« durchführen. Die Keytab wird direkt unter Linux geschrieben und muss nicht umständlich dorthin kopiert werden. Die Kommunikation mit dem AD erfolgt über Netzwerkprotokolle. Die Grundeinrichtung der Linux-Maschine wie in Abschnitt 15.3.3 beschrieben ist Voraussetzung. Das Listing 15.10 stellt diesen Vorgang dar und zeigt auch die resultierende Keytab-Datei.

Listing 15.10 *Linux-Domainjoin mit* `adcli` *schreibt Keytab*

```
root@lx02.ads:~# adcli join
Password for Administrator@ADS.EXAMPLE.COM:  P@ssw0rd
root@lx02.ads:~# klist -ke
Keytab name: FILE:/etc/krb5.keytab
KVNO Principal
---- --------------------------------------------------
   2 LX02$@ADS.EXAMPLE.COM (DEPRECATED:arcfour-hmac)
   2 LX02$@ADS.EXAMPLE.COM (aes128-cts-hmac-sha1-96)
   2 LX02$@ADS.EXAMPLE.COM (aes256-cts-hmac-sha1-96)
   2 host/LX02@ADS.EXAMPLE.COM (DEPRECATED:arcfour-hmac)
   2 host/LX02@ADS.EXAMPLE.COM (aes128-cts-hmac-sha1-96)
   2 host/LX02@ADS.EXAMPLE.COM (aes256-cts-hmac-sha1-96)
   2 host/lx02.ads.example.com@ADS.EXAMPLE.COM (DEPRECATED ➥
➥  :arcfour-hmac)
   2 host/lx02.ads.example.com@ADS.EXAMPLE.COM (aes128-cts ➥
➥  -hmac-sha1-96)
   2 host/lx02.ads.example.com@ADS.EXAMPLE.COM (aes256-cts ➥
➥  -hmac-sha1-96)
```

```
   2 RestrictedKrbHost/LX02@ADS.EXAMPLE.COM (DEPRECATED:
➥  arcfour-hmac)
   2 RestrictedKrbHost/LX02@ADS.EXAMPLE.COM (aes128-cts-
➥  hmac-sha1-96)
   2 RestrictedKrbHost/LX02@ADS.EXAMPLE.COM (aes256-cts-
➥  hmac-sha1-96)
   2 RestrictedKrbHost/lx02.ads.example.com@ADS.EXAMPLE.
➥  COM (DEPRECATED:arcfour-hmac)
   2 RestrictedKrbHost/lx02.ads.example.com@ADS.EXAMPLE.
➥  COM (aes128-cts-hmac-sha1-96)
   2 RestrictedKrbHost/lx02.ads.example.com@ADS.EXAMPLE.
➥  COM (aes256-cts-hmac-sha1-96)
root@lx02.ads:~#
```

`adcli` sollte bei CentOS die erste Wahl sein, wenn es um einfache Maschinenkonten und Host Keytabs geht. Für komplexere Anforderungen nutzt man besser das in Abschnitt 15.5.3 beschriebene `msktutil`.

15.5.3 Host Keytabs unter Linux mit msktutil verwalten

Auch mit `msktutil` lassen sich Keytabs für Maschinen- und für Dienste-Accounts verwalten. Bei CentOS stellt das EPEL-Repository dieses Tool zur Verfügung. Man installiert es am einfachsten per `dnf install msktutil`. Anhang C.5 behandelt weitere Möglichkeiten der Installation. Auch hier erfolgt die Kommunikation mit dem AD über das Netzwerk, vorausgesetzt die in Abschnitt 15.3.3 beschriebene Grundeinrichtung funktioniert.

Eine Host Keytab für den Maschinen-Account

Die einfachste Anwendung besteht darin, analog zu Abschnitt 15.5.1 einen Host Principal für das Linux-System zu erzeugen. Listing 15.11 zeigt dies am Beispiel der Maschine `lx02`. Die Einschränkung der Schlüsseltypen auf AES256 geschieht hier über die Option `--enctypes 0x10`. Als weiteres Resultat wird die Keytab-Datei `/etc/krb5.keytab` geschrieben.

Listing 15.11 *Unter Linux eine Keytab-Datei für den Host Principal erzeugen*

```
root@lx02.ads:~# kinit Administrator@ADS.EXAMPLE.COM
Password for Administrator@ADS.EXAMPLE.COM: P@ssw0rd
root@lx02.ads:~# msktutil create --enctypes 0x10
No computer account for lx02 found, creating a new one.
root@lx02.ads:~#
```

Das AD-Maschinenkonto `LX02$` wird hierbei automatisch erzeugt, falls es nicht bereits existiert. Das zugehörige Maschinenpasswort erhält einen zufälligen Wert. Dabei werden auch SPNs gesetzt. Anders als bei der `ktpass.exe`-Variante setzt `msktutil` defaultmäßig keinen UPN bei Maschinen-Accounts. Insgesamt sehen die für Kerberos wesentlichen Namensattribute am Objekt der `LX02$` anschließend wie folgt aus:

- `userPrincipalName`: nicht gesetzt
- `servicePrincipalName`: `host/lx02.ads.example.com` und `host/lx02`
- `sAMAccountName`: `lx02$`

Die mit `msktutil` erzeugte Keytab können Sie wie in Listing 15.12 dargestellt anzeigen und testen. Verglichen mit dem Resultat von `ktpass.exe` (Listing 15.9) fällt auf, dass hier auch der Wert des `sAMAccountName`-Attributes einen Eintrag in der Keytab erhalten hat – genau genommen sogar zwei, die sich lediglich in Groß- und Kleinschreibung unterscheiden: `lx02$@ADS.EXAMPLE.COM` und `LX02$@ADS.EXAMPLE.COM`. Außerdem stellt `host/lx02.ads.example.com@ADS.EXAMPLE.COM` hier offenbar keinen Client-Principal dar, was aufgrund des nicht gesetzten `userPrincipalName`-Attributs zu vermuten war und was die Meldung `Client 'host/lx02.ads.example.com@ADS.EXAMPLE.COM' not found in Kerberos database while getting initial credentials` bestätigt. Diese in reinen MIT-Kerberos-Umgebungen übliche Form einer Maschinen-Client-Identität ist in AD-Umgebungen weniger gebräuchlich. Man kann stattdessen `LX02$@ADS.EXAMPLE.COM` als Client-Principal verwenden, so wie das im Listing 15.12 getan wird[7].

Listing 15.12 *Mit* `msktutil` *erzeugte Keytab-Datei anzeigen und testen*

```
root@lx02.ads:~# klist -ke
Keytab name: FILE:/etc/krb5.keytab
KVNO Principal
---- --------------------------------------------------
   2 lx02$@ADS.EXAMPLE.COM (aes256-cts-hmac-sha1-96)
   2 LX02$@ADS.EXAMPLE.COM (aes256-cts-hmac-sha1-96)
   2 host/lx02@ADS.EXAMPLE.COM (aes256-cts-hmac-sha1-96)
   2 host/lx02.ads.example.com@ADS.EXAMPLE.COM (aes256-cts ⏎
➥  -hmac-sha1-96)
root@lx02.ads:~# kinit -k host/lx02.ads.example.com
kinit: Client 'host/lx02.ads.example.com@ADS.EXAMPLE.COM' ⏎
➥  not found in Kerberos database while getting initial  ⏎
➥  credentials
root@lx02.ads:~# kinit -k 'LX02$@ADS.EXAMPLE.COM'
root@lx02.ads:~# kvno -k /etc/krb5.keytab                 \
                        host/lx02.ads.example.com
host/lx02.ads.example.com@ADS.EXAMPLE.COM: kvno = 2,      ⏎
➥  keytab entry valid
root@lx02.ads:~# kdestroy
root@lx02.ads:~#
```

[7] Um alternativ auch den `userPrincipalName` zu setzen, sieht `msktutil` übrigens den Parameter `--upn` vor.

15.5.4 Keytabs für Service Accounts unter Linux mit msktutil verwalten

Client Keytabs

Mit `msktutil` lassen sich auch Client Keytabs verwalten. Im Folgenden werden zwei Szenarien betrachtet. Ziel ist jeweils, ein Benutzerobjekt mit dem Namen `techuser01` (Kerberos Principal: `techuser01@ADS.EXAMPLE.COM`) anzulegen und dessen Keys in der Datei `/etc/krb5.keytab.techuser01` zu hinterlegen.

Keytab für existierendes Benutzerobjekt mit bekanntem Passwort, ohne administrative Berechtigungen im AD

Ein übliches Szenario besteht darin, dass der Service Account im AD bereits angelegt wurde. Auf Linux-Seite gilt es nun die Keytab dazu zu erstellen. Ein initiales Passwort von `techuser01` soll dabei auf einen zufälligen Wert gesetzt werden. Gehen Sie wie in Abschnitt 15.4.3 beschrieben vor, um den Account für `techuser01` anzulegen. Dabei setzen Sie ein initiales Passwort (im Beispiel ist das »Start123«), das aber bei der ersten Anmeldung geändert werden muss. Unter Linux führt das in Listing 15.13 gezeigte Kommando die Passwortänderung durch und erzeugt dabei die Keytab.

Listing 15.13 *`msktutil` erzeugt eine Keytab für einen im AD vorbereiteten Service Account.*

```
root@lx02.ads:~# msktutil update                          \
                 --use-service-account                    \
                 --account-name techuser01                \
                 --old-account-password Start123          \
                 --keytab /etc/krb5.keytab.techuser01
root@lx02.ads:~#
```

Man benötigt dazu keine weiteren Berechtigungen im AD, da dort lediglich das Passwort geändert wird. Ein vorausgehendes `kinit` ist nicht erforderlich; die Angabe des alten Passwortes mittels `--old-account-password Start123` genügt. Ein wichtiger Punkt: An die Passwortrichtlinien des AD muss man sich hier halten. Das betrifft nicht die Komplexität, denn `msktutil` erzeugt zufällige Passwörter, die allen Komplexitätsanforderungen standhalten sollten. Allerdings ist die Richtlinie *Minimales Kennwortalter* (siehe Abbildung 15.16) hier zu beachten. Verwendet man ein existierendes Benutzerobjekt mit einem bekanntem Passwort, das nicht geändert werden muss, das aber vor kurzem geändert wurde, dann muss man gegebenenfalls warten, bis dieses wieder geändert werden darf. Das könnte eine Weile dauern, der Default ist ein Tag.

Mit administrativen AD-Berechtigungen

Möchte man sich nicht mit diesen Richtlinien herumschlagen, so kann man das Passwort einfach neu setzen, anstatt es zu ändern. Dazu benötigt man aber einen administrativen AD-Account (im folgenden Beispiel `Administrator`), für den man sich vorab mit `kinit` Tickets besorgen muss. Listing 15.14 beschreibt dieses Szenario, bei dem der zugehörige AD-Account `techuser01` auch automatisch erzeugt wird, falls er im Vorfeld noch nicht angelegt wurde. Hierbei ist wichtig, dass `msktutil`

nicht irgendwelche, sondern genau die per `kinit` bereitgestellten Credentials für die AD-Operationen benutzt, weswegen man hier die Option `--user-creds-only` angeben sollte.

```
root@lx02.ads:~# kinit Administrator
Password for Administrator@ADS.EXAMPLE.COM: P@ssw0rd
root@lx02.ads:~# msktutil update                              \
                 --use-service-account                        \
                 --account-name techuser01                    \
                 --user-creds-only                            \
                 --keytab /etc/krb5.keytab.techuser01
root@lx02.ads:~#
```

Listing 15.14 *`msktutil` erzeugt eine Keytab mit administrativen Redchten*

In beiden Szenarien ist es auch möglich, Service Accounts mit SPNs und zugehörigen Keytab-Einträgen zu verwalten, wobei diese im ersten Szenario ohne administrative Rechte AD-seitig gesetzt werden müssen. `msktutil` übernimmt diese SPNs dann in die erzeugte Keytab. Mit administrativen Credentials ist im zweiten Szenario auch das Erzeugen der SPNs über den `msktutil`-Parameter `--spn` möglich.

Tipp

Ein Cronjob mit dem Kommando `msktutil auto-update` führt dazu, dass die Maschinenpasswörter der Linux-Systeme regelmäßig (alle dreißig Tage) geändert werden. Das funktioniert auch mit Service Accounts.

15.6 Kerberos-Administration mit LDAP

In Abschnitt 15.4 haben Sie die Kerberos-Administration mit den Standardwerkzeugen des Active Directory kennengelernt: ADUC, der darin enthaltene *Attribut-Editor*, `setspn.exe` und `ktpass.exe` genügten dort, um Client- und Dienste-Principals und deren Eigenschaften zu verwalten. Im vorliegenden Abschnitt werden Sie sehen, wie man solche administrativen Aufgaben alternativ mit LDAP ausführen kann.

Der LDAP-Dienst auf `kdc01` und `kdc02` ist bereits kerberisiert. Das ist ein weiterer Pluspunkt für die AD-Umgebung, denn die bisher installierten OpenLDAP-Dienste der anderen Umgebungen sind noch nicht kerberisiert. Das wird erst in Kapitel 20 nachgeholt werden. Anhand von Active Directory können Sie also hier schon sehen, wie man LDAP-Operationen mit kerberisierten Clients durchführen kann.

Als Clients werden die OpenLDAP-Werkzeuge auf der CentOS-Maschine `lx01` eingesetzt. Folgende Schritte sind dort nötig, wenn Sie diesen Abschnitt nachvollziehen wollen:

Software: `openldap-clients cyrus-sasl-gssapi`

- Melden Sie sich auf der `lx01.ads.example.com` an und installieren Sie dort das Paket mit den OpenLDAP-Werkzeugen: `dnf install openldap-clients`.
- Kerberos wird im LDAP-Protokoll nicht direkt verwendet, sondern über das sogenannte GSS-API, das wiederum als Mechanismus innerhalb des SASL-Frameworks verwendet wird. Mehr Hintergrundwissen dazu bietet Abschnitt 19.7. An dieser Stelle ist es nur wichtig, das SASL/GSS-API-Modul zu installieren, um die LDAP-Clientbibliotheken unter Linux zu kerberisieren. Das entsprechende Paket `cyrus-sasl-gssapi` wird von `dnf` bereits mit installiert.
- Beziehen Sie noch ein Ticket für einen Benutzer, der im AD die nötigen Rechte hat: `kinit Administrator`. Diese Kerberos Credentials werden im Folgenden vorausgesetzt.

Von nun an können Sie – einen gültigen Credential Cache mit einem Administrator-Ticket vorausgesetzt – mit den folgenden Kommandos LDAP-Suchanfragen und andere Operationen an AD-Objekten von der `lx01` aus durchführen:

- `ldapsearch -h kdc01.ads.example.com -b "dc=ads,dc=example,dc=com" [LDAP-Suchfilter] [LDAP-Attribute]`
- `ldapadd -h kdc01.ads.example.com -f LDIF-Datei`
- `ldapmodify -h kdc01.ads.example.com -f LDIF-Datei`
- `ldapdelete -h kdc01.ads.example.com "DN des AD-Objektes"`

15.6.1 LDAP-Suchen im AD

Listing 15.15 zeigt als Beispiel eine LDAP-Suche mit `ldapsearch`. Durch den *Suchfilter* `cn="Max Mustermann"` wird eine Suche nach dem Objekt von Max Mustermann ausgeführt. Außerdem wird die LDAP-Suche auf folgende Attribute eingeschränkt: `objectClass`, `cn`, `sn`, `givenName`, `displayName`, `samaccountname`, `userPrincipalName`, `unicodePwd` und `msDS-KeyVersionNumber`. Wie Sie sehen, ähnelt der Nutzereintrag im AD kaum seinem Äquivalent im OpenLDAP-Verzeichnis aus Abschnitt 13.4 (Listing 13.10 oder 13.13). Obwohl es sich bei beiden Nutzerdatenbanken um LDAP-Verzeichnisse handelt, verwenden sie unterschiedliche Objektklassen und Attributnamen, um Benutzerobjekte darzustellen.

```
root@lx01.ads:~# ldapsearch -LLL                                    \
                 -h kdc01.ads.example.com                           \
                 -b dc=ads,dc=example,dc=com                        \
                 cn="Max Mustermann"                                \
                 objectClass cn sn givenName                        \
                 displayName samaccountname                         \
                 userPrincipalName unicodePwd                       \
                 msDS-KeyVersionNumber
SASL/GSS-SPNEGO authentication started
SASL username: Administrator@ADS.EXAMPLE.COM
SASL SSF: 256
SASL data security layer installed.
dn: CN=Max Mustermann,CN=Users,DC=ads,DC=example,DC=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: Max Mustermann
sn: Mustermann
givenName: Max
displayName: Max Mustermann
sAMAccountName: maxm
userPrincipalName: maxm@ads.example.com

# refldap://DomainDnsZones.ads.example.com/
 DC=DomainDnsZones,DC=ads,DC=example,DC=com

# refldap://ForestDnsZones.ads.example.com/
 DC=ForestDnsZones,DC=ads,DC=example,DC=com

# refldap://ads.example.com/CN=Configuration,
 DC=ads,DC=example,DC=com

root@lx01.ads:~#
```

Listing 15.15 *Test des kerberisierten LDAP-Clients: Suche nach dem Benutzerobjekt von Max Mustermann*

Nicht alle Attribute werden im Suchergebnis aus Listing 15.15 angezeigt. Beispielsweise ist das Attribut `unicodePwd` nicht zu sehen. `unicodePwd` dient der Speicherung des Passwort-Hash und kann aus Sicherheitsgründen nicht über LDAP gelesen werden.

`unicodePwd` *nicht lesbar*

15.6.2 Ein Benutzerobjekt anlegen

Sie können Active Directory mit LDAP nicht nur durchsuchen, auch Änderungen am Datenbestand sind möglich. Hier soll nun ein Benutzerobjekt für Erika Musterfrau angelegt werden, diesmal aber mit dem Kommando `ldapadd` unter Linux.

Dazu müssen Sie als Erstes eine LDIF-Beschreibung des Benutzerobjektes erzeugen. Nicht alle Attribute, die ein Nutzerobjekt im AD haben kann, sind dazu nötig, es genügt eine Auswahl der wesentlichen Attribute. Ein wesentliches Attribut ist `unicodePwd`. Dieses kann man mit LDAP zwar nicht auslesen, es ist aber über LDAP schreibbar.

`unicodePwd` *schreibbar*

Beim Schreiben von `unicodePwd` muss man das Anwenderpasswort in das richtige Format bringen. Wie der Attributname vermuten lässt, muss das Passwort als *Unicode String* vorliegen. Das ist aber noch nicht alles, man muss das Passwort zusätzlich noch in doppelte Anführungszeichen (`"…"`) einpacken, die nicht Bestandteil des Passworts sind. Beispielsweise muss man das Klartextpasswort `P@ssw0rd` zuerst zu `"P@ssw0rd"` ergänzen und dann diesen String nach Unicode konvertieren. Für das LDIF-Format, das ja nur 7-Bit-ASCII versteht, muss der Unicode String noch nach BASE64 gewandelt werden. Das Ergebnis der Konvertierung von »`P@ssw0rd`« lautet »`IgBQAEAAcwBzAHcAMAByAGQAIgA=`«. Listing 15.16 zeigt ein Skript in der Sprache Python, das die beschriebene Konvertierung durchführt.

Listing 15.16 *Das Python-Skript* `ad_unicodepwd` *konvertiert Klartextpasswörter in AD-kompatible Unicode Strings.*

```
#!/usr/bin/env python3
import sys
import base64
if len(sys.argv) != 2:
    print ('usage: ' + sys.argv[0] + ' <password>')
    sys.exit()
password = sys.argv[1]
quotedPassword = '"' + password + '"'
unicodePwd = quotedPassword.encode('utf_16_le')
print ('unicodePwd:: ' +
       base64.b64encode(unicodePwd).decode('utf8'))
```

Wenn Sie das Skript aus Listing 15.16 beispielsweise unter dem Namen `ad_unicodepwd` abspeichern und die Datei ausführbar machen (`chmod +x ad_unicodepwd`), dann können Sie die Passwortkonvertierung wie in Listing 15.17 dargestellt durchführen.

Listing 15.17 *Aufruf von* `ad_unicodepwd`

```
root@lx01.ads:~# ./ad_unicodepwd P@ssw0rd
unicodePwd:: IgBQAEAAcwBzAHcAMAByAGQAIgA=
root@lx01.ads:~#
```

Die anderen Attribute für das neue Nutzerobjekt für Erika Musterfrau sind einfacher zu erzeugen.

```
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
instanceType: 4
displayName: Erika Musterfrau
name: Erika Musterfrau
userAccountControl: 512
sAMAccountName: erim
userPrincipalName: erim@ADS.EXAMPLE.COM
unicodePwd:: IgBQAEAAcwBzAHcAMAByAGQAIgA=
pwdLastSet: 0
```

Listing 15.18 *LDIF-Darstellung des neuen Nutzerobjektes*

Listing 15.18 stellt eine LDIF-Datei dar, die alle notwendigen Attribute enthält. Um die so in LDIF beschriebene Benutzerin zum Active Directory hinzuzufügen, verwenden Sie ein Kommando wie `ldapadd -h kdc01 -f listing-15.18.ldif`.

Die LDIF-Beschreibung enthält das unverschlüsselte[8] Passwort. Daher muss das Einspielen der LDIF-Datei verschlüsselt erfolgen, AD lässt eine Änderung am Attribut `unicodePwd` über unverschlüsselte LDAP-Operationen nämlich gar nicht zu. Diese Verschlüsselung der LDAP-Daten wird hier durch Verwendung von SASL/GSS-API oder SASL/GSS-SPNEGO (und damit Kerberos) gewährleistet.

15.6.3 Diensteobjekte anlegen

So wie in Abschnitt 15.6.2 können Sie mit LDAP auch Dienste-Accounts anlegen. Dazu müssen Sie eine LDIF-Datei wie die in Listing 15.18 benutzen, die zusätzlich noch das Attribut `servicePrincipalName` enthält.

[8] Die BASE64-Darstellung wirkt zwar kryptisch, stellt aber keine Verschlüsselung dar!

15.6.4 Maschinenobjekte anlegen

Das Erzeugen eines Active-Directory-Objektes für die zweite Linux-Maschine `lx02.ads.example.com` kann nach dem gleichen Muster erfolgen wie die Erzeugung des Nutzerobjektes in Abschnitt 15.6.2.

Auch ein Maschinen-Account benötigt ein Passwort, das möglichst komplex und zufällig erzeugt sein sollte. Auch dieses generierte Passwort kann man mit einem Tool wie `ad_unicodepwd` aus Listing 15.16 AD-kompatibel machen. Das Listing 15.19 stellt die Erzeugung des Maschinenpasswortes für die `lx02` dar. Damit können Sie eine LDIF-Datei erzeugen, die alle relevanten Attribute des Maschinenobjektes enthält. Listing 15.20 zeigt ein solches LDIF.

Listing 15.19 *Erzeugung eines Maschinenpasswortes für die* `lx02`

```
root@lx01.ads:~# tr -cd '[:alnum:]' < /dev/urandom |       \
                                        head -c 25; echo
hNJvEb2V5OYZ7PAstqQQwJah5
root@lx01.ads:~# ./ad_unicodepwd hNJvEb2V5OYZ7PAstqQQwJah5
unicodePwd:: IgBoAE4ASgB2AEUAYgAyAFYANQAwAFkAWgA3AFAAQQBzA ➥
➥  HQAcQBRAFEAdwBKAGEAaAA1ACIA
root@lx01.ads:~#
```

Listing 15.20 *LDIF-Darstellung des neuen Maschinenobjektes*

```
dn: CN=lx02,CN=Computers,DC=ADS,DC=EXAMPLE,DC=COM
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
objectClass: computer
cn: lx02
instanceType: 4
displayName: lx02$
name: lx02
userAccountControl: 4096
sAMAccountName: lx02$
unicodePwd:: IgBoAE4ASgB2AEUAYgAyAFYANQAwAFkAWgA3AFAAQQBzA
 HQAcQBRAFEAdwBKAGEAaAA1ACIA
userPrincipalName: host/lx02.ads.example.com@ADS.
 EXAMPLE.COM
msDS-SupportedEncryptionTypes: 24
```

Anders als das Benutzerobjekt gehört das Maschinenobjekt auch der Objektklasse `computer` an. Das LDIF aus Listing 15.20 setzt auch gleich das Attribut `msDS-SupportedEncryptionTypes` und aktiviert damit die AES-

Verschlüsselung für die lx02 (siehe Abschnitt 15.4.4). Sie können das Maschinenobjekt mit ldapadd einspielen.

Für die Funktionalität als Kerberos Service-Principal fehlen dem LX02$-Objekt noch die servicePrincipalName-Attribute. Im Falle der lx01 wurden diese unter Windows mit setspn.exe -R gesetzt. Alternativ dazu können Sie das LDIF aus Listing 15.21 mit ldapmodify unter Linux für die SPNs der lx02 anwenden.

Listing 15.21
Die SPNs setzen

```
dn: CN=lx02,CN=Computers,DC=ADS,DC=EXAMPLE,DC=COM
changetype: modify
add: servicePrincipalName
servicePrincipalName: host/lx02.ads.example.com
servicePrincipalName: host/lx02
```

Keytab mit ktutil

Für die Kerberisierung der Maschine lx02 fehlt aber noch ein Schritt: Das Maschinenpasswort aus Listing 15.19 muss auf der lx02 in der Keytab-Datei /etc/krb5.keytab hinterlegt werden. Das Erzeugen der Keytab bietet hier nochmals die Möglichkeit, das MIT Utility ktutil näher kennenzulernen (siehe auch Abschnitt 9.6.2). Voraussetzung ist hierfür lediglich die Kenntnis des Maschinenpasswortes im Klartext. Listing 15.22 stellt den Vorgang dar.

Listing 15.22
Erzeugung der Keytab mit ktutil

```
root@lx02.ads:~# kinit Administrator
Password for Administrator@ADS.EXAMPLE.COM: P@ssw0rd
root@lx02.ads:~# kvno host/lx02.ads.example.com
host/lx02.ads.example.com@ADS.EXAMPLE.COM: kvno = 1
root@lx02.ads:~# ktutil
ktutil: addent -password -p host/lx02.ads.example.com ↵
➥              -k 1 -e aes256-cts
Password for host/lx02.ads.example.com@ADS.EXAMPLE.COM: ↵
➥  hNJvEb2V50YZ7PAstqQQwJah5
ktutil: wkt /etc/krb5.keytab
ktutil: quit
root@lx02.ads:~# kinit -kt /etc/krb5.keytab                \
                  host/lx02.ads.example.com
root@lx02.ads:~# kvno -k /etc/krb5.keytab                  \
                  host/lx02.ads.example.com
host/lx02.ads.example.com@ADS.EXAMPLE.COM: kvno = 1, ↵
➥  keytab entry valid
root@lx02.ads:~#
```

Als Erstes müssen Sie die Versionsnummer (KVNO) feststellen, was in Listing 15.22 mit dem ersten Aufruf von kvno erfolgt. Dazu benötigen Sie einen frischen Credential Cache, den das vorangehende kinit-Kommando

erzeugt. Danach starten Sie `ktutil` und erzeugen mit `addent` einen neuen Eintrag. Die Option `-password` legt fest, dass der Schlüssel in Form eines Passwortes angegeben werden soll. Dann müssen Sie noch den Namen des Principals (`-p`), die KVNO (`-k`) und den Verschlüsselungstyp (`-e`) festlegen. Anschließend werden Sie nach dem Passwort gefragt und können die Keytab mit dem Kommando `wkt` schreiben. Die letzten beiden Aufrufe von `kinit -kt` und `kvno -k` dienen wiederum dem Test der erzeugten Keytab-Datei.

Achtung

Die beschriebene Vorgehensweise mit `ktutil` ist durchaus fehleranfällig. Vertippt man sich beim Passwort, dann wird ein falscher Schlüssel in die Keytab geschrieben. Auch durch eine falsche KVNO wird die Keytab eventuell unbrauchbar. Aber auch der Name des Principals, den man mit `-p` angibt, kann als Salt String mit in die Schlüsselgenerierung einfließen. Der Principal-Name muss für Host Principals folgende Form haben: `host/`*`host.ad-domain`*`@`*`REALM`*. Dabei ist *`host`* der kurze Hostname des Rechners und *`ad-domain`* der kleingeschriebene Realm-Name der AD-Domäne. Fazit: Testen Sie erzeugte Keytabs immer mit `kvno -k`.

15.7 Weitere Werkzeuge

Für die Kerberos-Administration in heterogenen AD-Domänen mit Unix- und Linux-Mitgliedern gibt es eine Reihe weiterer Möglichkeiten:

- Die Open-Source-Software *Samba* bietet im `net`-Kommando eine Reihe von Subkommandos, die man für die Verwaltung von Nutzer- und Host Principals einsetzen kann. Damit lassen sich auch Keytabs pflegen, allerdings mit gewissen Einschränkungen: So kann das `net`-Kommando nicht für die Verwaltung von allgemeinen Diensteobjekten benutzt werden. Mehr zu Samba (als Dateidienst) werden Sie in Abschnitt 23.1 erfahren. Samba als alternative AD-Infrastruktur folgt in Kapitel 16.
- Kommerzielle Tools zur *AD-Integration* wie *DirectControl* von *Centrify*, *Quest Authentication Services (QAS)* von *Quest Software* oder *PowerBroker Identity Services* von *BeyondTrust* (ehemals *Likewise*) bieten ebenfalls eine Reihe von Möglichkeiten, um die Kerberos-Verwaltung in AD-Umgebungen mit integrierten Unix- und Linux-Systemen zu vereinfachen (siehe auch [27]).