3 Kerberos aus Anwendersicht

Nachdem Sie in den Kapiteln 1 und 2 einen Überblick über die Netzwerkauthentisierung mit Kerberos erhalten haben, werden Sie hier nun die Kerberos-Praxis aus der Sicht eines Beispielnutzers kennenlernen.

Alle Clients und Dienste in den Beispielen unterstützen Kerberos bereits, sie sind kerberisiert. *Wie Sie eine solche kerberisierte Umgebung selbst einrichten können, erfahren Sie ab Teil II dieses Buches. Das vorliegende Kapitel soll Ihnen zunächst einen Einblick in die Anwenderseite von Kerberos verschaffen.*

3.1 Die Beispielumgebung

Der Anwender der Beispielumgebung heißt Max Mustermann und arbeitet an einer Linux-Workstation (Distribution: CentOS 8) mit dem Namen `lx01`. Der Name des Kerberos Realm ist `EXAMPLE.COM` und entspricht dem Namen der DNS-Domäne (`example.com`). Der Kerberos-Dienst der Beispielumgebung läuft auf einem Server mit dem Namen `kdc01` (ebenfalls eine CentOS-Maschine).

Neben seiner lokalen Workstation `lx01` meldet sich Herr Mustermann hin und wieder auch per Secure Shell auf einer weiteren Maschine mit dem Namen `lx02` an. Außer diesem Secure-Shell-Dienst gibt es noch einen LDAP-Verzeichnisdienst (OpenLDAP) und einen Webserver (Apache). Der Webserver läuft auf der `lx02`, der Verzeichnisdienst auf der `kdc01`. Im DNS ist ein Alias für `lx02.example.com` eingerichtet, sodass Herr Mustermann den Webserver unter der URL *http://www.example.com* ansprechen kann. Abbildung 3.1 zeigt diese Beispielumgebung.

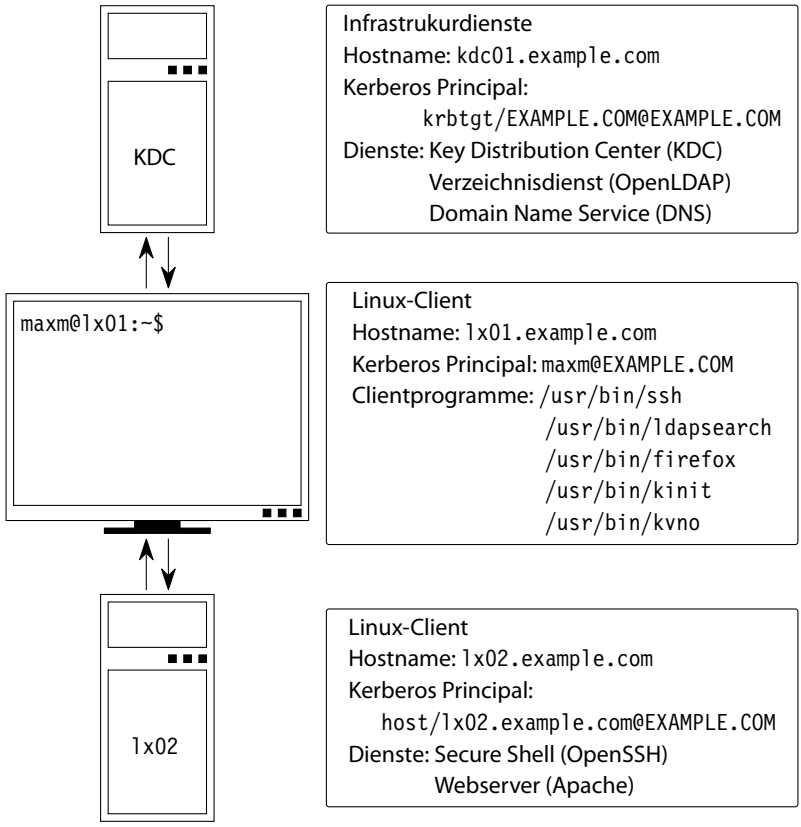


Abb. 3.1
Die Beispielumgebung

3.2 Lokale Anmeldung

Der Anwender Max Mustermann meldet sich lokal am Linux-Arbeitsplatz `lx01` an. Wie in Abschnitt 2.1 beschrieben, muss er sich dazu gegenüber der `lx01` authentisieren, indem er dem Login-Programm seinen Nutzernamen (`maxm`) und sein Passwort (»P@ssw0rd«) übergibt. Das Listing 3.1 zeigt die lokale Anmeldung an der nichtgrafischen Linux-Konsole, die Abbildung 3.2 das grafische Äquivalent.

Listing 3.1
Lokale Anmeldung an der Linux-Konsole

```
CentOS Linux 8 (Core)
Kernel 4.18.0-193.14.2.el8_2.x86_64 on an x86_64

lx01 login: maxm
Password: P@ssw0rd
maxm@lx01:~$
```

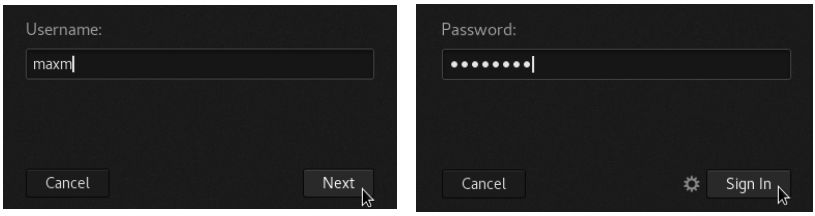


Abb. 3.2
Lokale Anmeldung mit dem grafischen Login-Programm GNOME Display Manager (GDM)

Die Passwortüberprüfung an der `lx01` funktioniert bereits über Kerberos. Man sagt auch: Das Anmeldeprogramm ist *kerberisiert*. Max Mustermann hat dabei bereits einmal sein Passwort angegeben; im Sinne des in Abschnitt 2.5 beschriebenen *Single Sign-on (SSO)* sollte er damit netzwerkweit angemeldet sein. Beim Zugriff auf die Beispieldienste wird sich Herr Mustermann durch Kerberos Tickets authentisieren. Sollte er nochmals nach einem Passwort gefragt werden, dann würde das bedeuten, dass eine Fehlkonfiguration vorliegt.

3.3 Der Credential Cache

Das kerberisierte Login-Programm legt die Kerberos Tickets, die es im Rahmen der Passwortüberprüfung erhalten hat, in einem sogenannten *Credential Cache* ab. Auch `maxm` hat bei seiner Anmeldung einen Credential Cache erhalten.

Das Kerberos-Kommando `klist` dient zur Anzeige des Inhalts von Credential Caches. In Listing 3.2 ist ein Aufruf von `klist` dargestellt, den `maxm` direkt nach der Anmeldung an der `lx01` eingegeben hat. Dieser Aufruf des `klist`-Kommandos erfolgt dabei ohne weitere Kommandozeilenoptionen.

```
maxm@lx01:~$ klist
Ticket cache: FILE:/tmp/krb5cc_10000_zGOviZ
Default principal: maxm@EXAMPLE.COM

Valid starting      Expires             Service principal
06/22/2021 17:32:59 06/23/2021 03:32:59  krbtgt/EXAMPLE.
➥  COM@EXAMPLE.COM
       renew until 06/23/2021 17:32:57
maxm@lx01:~$
```

Listing 3.2
Ausgabe des Kommandos `klist`

Wie Sie in Listing 3.2 sehen können, liegt der Kerberos-v5-Cache in der Datei `/tmp/krb5cc_10000_zGOviZ`. `10000` ist dabei die numerische Benutzer-ID von `maxm`, `zGOviZ` ist ein zufällig gewählter Anhang an den Dateinamen. Eine weitere Information in der `klist`-Ausgabe ist die Angabe des `Default principal`. Der gibt Herrn Mustermanns Kerberos-

Identität an, also seinen Principal-Namen `maxm@EXAMPLE.COM`. Weiter unten werden dann die Kerberos Tickets gelistet. Zum jetzigen Zeitpunkt (direkt nach der lokalen Anmeldung) liegt nur eines für den Service `krbtgt/EXAMPLE.COM@EXAMPLE.COM` vor. Dies ist das sogenannte *Ticket-Granting Ticket (TGT)*.

Ticket-Granting Ticket

In den Ausgaben von `klist` in Listing 3.2 sind noch weitere Angaben zu diesem Ticket zu sehen: In der Spalte `Valid starting` steht der Zeitpunkt, ab wann das Ticket gültig ist, in `Expires`, wann es abläuft. Der Angabe nach `renew until` kann man entnehmen, wie lange es erneuerbar ist (Abschnitt 6.4 behandelt Ticket-Erneuerung im Detail).

In Listing 3.3 sehen Sie die Auswirkung der zusätzlichen Option `-f`. Damit zeigt `klist` zusätzlich auch die gesetzten *Ticket Flags* an. Im vorliegenden Beispiel sind das die Flags `FRIA`; sie bedeuten im Einzelnen: ***F**orwardable*, ***R**enewable*, ***I**nitial* und *Pre-**A**uthenticated*. Was es mit diesen Flags genau auf sich hat, werden Sie in Abschnitt 6.2 sehen.

Listing 3.3 *Mit der Kommandozeilenoptionen* `-f` *zeigt* `klist` *zusätzliche* Ticket Flags *an.*

```
maxm@lx01:~$ klist -f
Ticket cache: FILE:/tmp/krb5cc_10000_zGOviZ
Default principal: maxm@EXAMPLE.COM

Valid starting      Expires             Service principal
06/22/2021 17:32:59 06/23/2021 03:32:59  krbtgt/EXAMPLE.
➥  COM@EXAMPLE.COM
       renew until 06/23/2021 17:32:57, Flags: FRIA
```

3.4 Anmeldung an Netzwerkdiensten

Herr Mustermann hat also bereits alles Nötige, um auf kerberisierte Netzwerkdienste zugreifen zu können. Beispielsweise kann er eine Suchanfrage an den LDAP-Verzeichnisdienst auf dem Server `kdc01` stellen. Dieser Zugriff und die resultierende Änderung am Credential Cache sind in Listing 3.4 dargestellt: Herr Mustermann benutzt das OpenLDAP-Kommando[1] `ldapsearch`, um nach einem LDAP-Eintrag mit einem `uid`-Attribut zu suchen, das den Wert `maxm` hat. Von diesem Verzeichniseintrag fragt er die Werte der beiden Attribute `uidNumber` und `gidNumber` ab. Mit der Option `-h` gibt er den Hostnamen des LDAP-Servers (`kdc01`) an. Die Optionen `-QLLL` haben nur den Zweck, das Suchergebnis übersichtlicher darzustellen. Die genauen Details der LDAP-Suche sind an dieser Stelle

[1] Die OpenLDAP-Kommandos dienen hier als Beispiel für Netzwerkclients und werden auch im weiteren Verlauf des Buches benötigt. Leser ohne LDAP-Kenntnisse finden einen Schnelleinstieg in Anhang A.

unwesentlich. Hier ist nur wichtig, dass es sich um einen Client-Server-Zugriff im Sinne von Abschnitt 2.1.3 handelt.

Anschließend hat sich der Credential Cache verändert. Die Ausgabe von `klist -f` in Listing 3.4 zeigt, dass der Credential Cache nun ein zusätzliches *Service Ticket* enthält. Dieses ist für den Dienste-Principal `ldap/kdc01.example.com@EXAMPLE.COM` ausgestellt worden.

Service Ticket

```
maxm@lx01:~$ ldapsearch -h kdc01 -QLLL                          \
                        uid=maxm uidNumber gidNumber
dn: uid=maxm,ou=people,dc=example,dc=com
uidNumber: 10000
gidNumber: 10000

maxm@lx01:~$ klist -f
Ticket cache: FILE:/tmp/krb5cc_10000_zGOviZ
Default principal: maxm@EXAMPLE.COM

Valid starting      Expires             Service principal
06/22/2021 17:32:59 06/23/2021 03:32:59  krbtgt/EXAMPLE.
➥  COM@EXAMPLE.COM
       renew until 06/23/2021 17:32:57, Flags: FRIA
06/22/2021 17:34:07 06/23/2021 03:32:59  ldap/kdc01.
➥  example.com@EXAMPLE.COM
       renew until 06/23/2021 17:32:57, Flags: FRAT
maxm@lx01:~$
```

Listing 3.4
Netzwerkzugriff am Beispiel des LDAP-Verzeichnisdienstes

Das Listing 3.5 zeigt einen weiteren kerberisierten Netzwerkzugriff, diesmal am Beispiel der Secure Shell (SSH).

```
maxm@lx01:~$ ssh lx02.example.com
Last login: Fri Aug 21 14:28:07 2020 from 10.1.2.111
maxm@lx02:~$ klist -f
klist: No credentials cache found (filename: /tmp/
➥  krb5cc_10000)
maxm@lx02:~$ logout
Connection to lx02.example.com closed.
maxm@lx01:~$ klist -f
Ticket cache: FILE:/tmp/krb5cc_10000_zGOviZ
Default principal: maxm@EXAMPLE.COM

Valid starting      Expires             Service principal
06/22/2021 17:32:59 06/23/2021 03:32:59  krbtgt/EXAMPLE.
➥  COM@EXAMPLE.COM
       renew until 06/23/2021 17:32:57, Flags: FRIA
```

Listing 3.5
Netzwerkzugriff am Beispiel einer Secure-Shell-Anmeldung. Auf der `lx02` *liegen keine Kerberos Tickets vor. Der Credential Cache auf der* `lx01` *erweitert sich durch die SSH-Sitzung um ein Host Ticket für die* `lx02`*.*

```
06/22/2021 17:34:07 06/23/2021 03:32:59  ldap/kdc01.          ➥
➥  example.com@EXAMPLE.COM
       renew until 06/23/2021 17:32:57, Flags: FRAT
06/22/2021 17:35:53 06/23/2021 03:32:59  host/lx02.example    ➥
➥  .com@EXAMPLE.COM
       renew until 06/23/2021 17:32:57, Flags: FRAT
maxm@lx01:~$
```

Max Mustermann meldet sich über SSH an der Workstation `lx02` an. Dort kann er beliebige Linux-Kommandos ausführen. Er gibt das Kommando `klist -f` ein, muss aber feststellen, dass ihm auf der `lx02` keine Kerberos Tickets zur Verfügung stehen. Er beendet die Remote-Sitzung auf der `lx02` mit dem Kommando `exit`. Zurück auf der `lx01` führt er das Kommando `klist -f` aus und stellt fest, dass sich sein Credential Cache um ein *Host Ticket* für `host/lx02.example.com@EXAMPLE.COM` erweitert hat.

Host Ticket

Das Prinzip ist also das folgende: Netzwerkclients beziehen bei Bedarf Kerberos Tickets für ihre Dienste. Der Credential Cache erweitert sich damit um immer mehr Dienste-Tickets. Alles, was ein Clientprogramm dazu benötigt, ist ein gültiges TGT.

3.5 Delegation

Die SSH-Anmeldung im letzten Beispiel hat noch einen Schönheitsfehler: Auf der entfernten Maschine `lx02` sind nach dem SSH-Login zunächst keine Kerberos Tickets vorhanden. Die könnten dort aber auch praktisch sein, beispielsweise dann, wenn Herr Mustermann eine LDAP-Anfrage, wie sie in Listing 3.4 dargestellt ist, auch von der `lx02` aus stellen möchte. Eine Möglichkeit, um dort an Kerberos Tickets zu gelangen, ist das Kerberos-Kommando `kinit`, das Sie in Abschnitt 3.7 kennenlernen werden. Herr Mustermann müsste dazu nochmals sein Passwort eingeben, was aber nicht der Idee des Single Sign-on entspricht. Schöner wäre es, wenn man die Kerberos Tickets einfach »mitnehmen« könnte.

Wie Sie aus Abschnitt 2.3 wissen, enthält Kerberos v5 Mechanismen zur *Delegation*. Dabei können Dienste die Identität ihrer Clients gegenüber anderen Diensten annehmen. Eine Delegationsvariante ist das *Ticket Forwarding*, bei dem der Client ein Ticket-Granting Ticket an den Service schickt – also genau das, was Herr Mustermann hier benötigt. Vorher aber noch eine Warnung:

Achtung

Bitte beachten Sie, dass Ticket Forwarding ein sicherheitsrelevanter Vorgang ist. Gerade weil Sie einem Dienst die Möglichkeit geben, Ihre Identität anzunehmen, müssen Sie diesem Dienst (und dem Server, auf dem er läuft) besonders vertrauen können. Benutzen Sie die Möglichkeit des Ticket Forwarding nur, wenn dieses Vertrauen besteht.

Abgesehen davon ist ein entfernter SSH-Login mit gleichzeitiger Ticket-Weiterleitung natürlich sehr praktisch und erhöht den Komfort. Den OpenSSH-Client kann man zum Ticket Forwarding bewegen, indem man die Konfigurationsoption `GSSAPIDelegateCredentials` auf `yes` setzt. Listing 3.6 stellt denselben Vorgang wie Listing 3.5 nochmals dar, diesmal wird aber ein TGT weitergeleitet.

```
maxm@lx01:~$ ssh -o GSSAPIDelegateCredentials=yes          \
                 lx02.example.com
Last login: Fri Aug 21 14:33:29 2020 from lx01.example.com
maxm@lx02:~$ klist -f
Ticket cache: FILE:/tmp/krb5cc_10000_tYAs5NLrnP
Default principal: maxm@EXAMPLE.COM

Valid starting      Expires             Service principal
06/22/2021 17:37:28 06/23/2021 03:32:59  krbtgt/EXAMPLE.
➥  COM@EXAMPLE.COM
       renew until 06/23/2021 17:32:57, Flags: FfRAT
maxm@lx02:~$ logout
Connection to lx02.example.com closed.
maxm@lx01:~$
```

Listing 3.6 *Secure-Shell-Anmeldung mit Ticket Forwarding. Anders als in Listing 3.5 liegen auf der* `lx02` *jetzt Kerberos Tickets vor.*

Sie können das an dem auf der entfernten `lx02` ausgeführten Kommando `klist -f` erkennen: Anders als in Listing 3.5 sind die Tickets nun auch auf der entfernten Maschine vorhanden. Beachten Sie auch die Flags (`FfRAT`): Dem weitergeleiteten TGT fehlt das `I`-Flag (Initial), dafür ist nun aber zusätzlich das `f`-Flag (*forwarded*) gesetzt. Das `F`-Flag des TGT im ursprünglichen Credential Cache (also dem auf der `lx01`) ist übrigens notwendig, damit das Ticket Forwarding so wie hier beschrieben funktioniert.

Anstelle von `-o GSSAPIDelegateCredentials=yes` bietet OpenSSH auch die kurze Kommandozeilenoption `-K`, um das Ticket Forwarding zu erlauben, und `-k`, um es zu verbieten. Anwender wie Herr Mustermann können die Konfigurationsoption auch, wie in Listing 3.7 dargestellt, in der Konfigurationsdatei `~/.ssh/config` für ausgewählte Zielhosts fest konfigurieren.

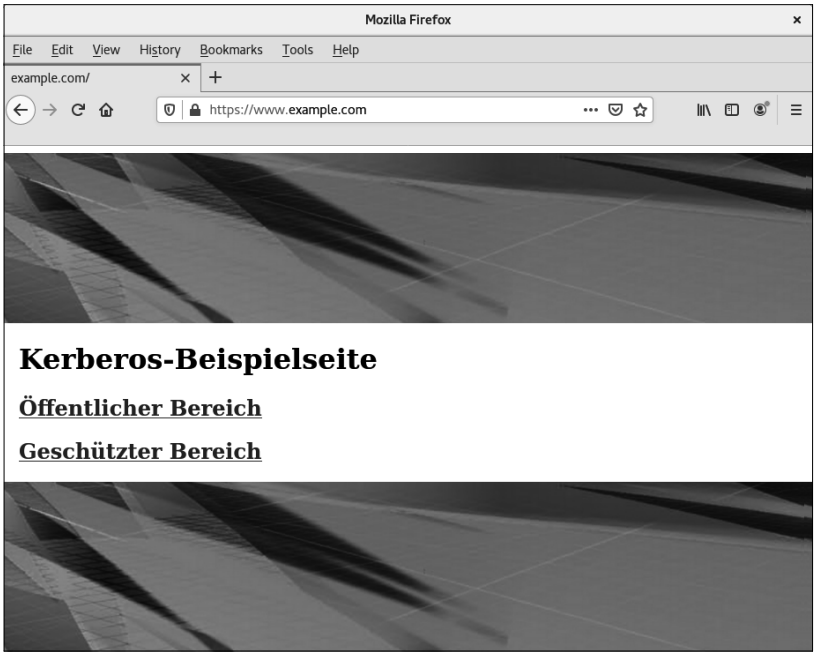
Listing 3.7
`~/.ssh/config`

```
Host lx02.example.com
     GSSAPIDelegateCredentials yes
```

In Abschnitt 22.1.3 werden weitere Konfigurationsmöglichkeiten der OpenSSH in Zusammenhang mit Kerberos-Authentisierung behandelt.

3.6 Eine Demo-Webseite

Abb. 3.3
Die Startseite bietet zwei Links an: »Öffentlicher Bereich« und »Geschützter Bereich«. Nach dem Klick auf den Link »Geschützter Bereich« wird Herr Mustermann mit einer Passwortabfrage konfrontiert (Abbildung 3.4).



Die Beispielumgebung beinhaltet auch einen Apache-Webserver. Dieser läuft auf der `lx02` und ist unter der URL `http://www.example.com` erreichbar. Die Hauptseite enthält zwei Links mit den Namen *Öffentlicher Bereich* und *Geschützter Bereich* (siehe Abbildung 3.3). Auf Letzteren darf man nur nach erfolgreicher Kerberos-Authentisierung zugreifen. Um zusätzlich den Netzwerkverkehr abhörsicher zu machen, ist die geschützte Seite nur über SSL erreichbar.

Hinter dem Hyperlink *Geschützter Bereich* verbirgt sich die URL `https://www.example.com/protected/index.cgi`. Das ist eine beispielhafte Anwendung in Form eines CGI-Skripts, das die Kerberos-Informationen der zugreifenden Clients anzeigen und stellvertretend für diese auf andere Netzwerkdienste zugreifen kann – Letzteres allerdings nur, wenn der Webbrowser Kerberos Tickets weiterleitet.

So weit jedenfalls die Idee. Wenn Herr Mustermann mit seinem Firefox-Webbrowser auf *Geschützter Bereich* klickt, so wird er trotz seiner Kerberos Tickets mit der in Abbildung 3.4 dargestellten Fehlerseite konfrontiert.

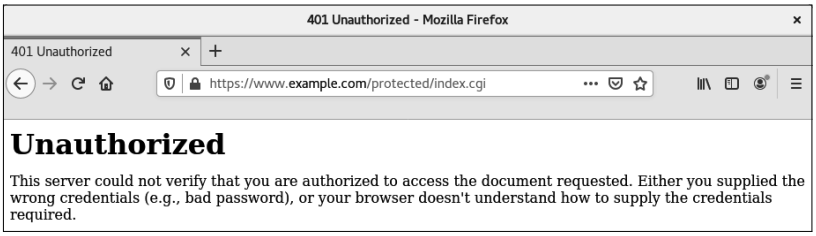


Abb. 3.4
So richtig funktioniert das SSO wohl noch nicht.

So wie der Firefox in diesem Beispiel verhalten sich die meisten Webbrowser, wenn sie auf eine mit Kerberos geschützte Seite treffen. Auch wenn sie Kerberos softwareseitig unterstützen. Das tun zwar bei Weitem nicht alle Browser, wohl aber der Firefox im Beispiel. Auch Edge oder der Internet Explorer (IE) von Microsoft kennen Kerberos und würden sich im Beispielszenario ähnlich verhalten.

Das Problem ist also nicht etwa eine fehlende Kerberos-Unterstützung, sondern vielmehr, dass diese noch nicht freigeschaltet ist. Dazu muss man in der Firefox-Option *network.negotiate-auth.trusted-uris* sämtliche URLs auflisten, gegenüber denen eine Kerberos-Authentisierung stattfinden soll[2]. Warum das so ist, erfahren Sie in Kapitel 24.

Um den Firefox zur Verwendung von Kerberos für eine bestimmte Site zu bewegen, gibt Herr Mustermann zunächst die URL *about:config* ein. Damit erhält er Zugriff auf alle Konfigurationsparameter des Webbrowsers. Er sollte genau wissen, was er tut, wenn er hier Änderungen vornimmt. Neuere Firefox-Versionen weisen auf diesen Umstand mit einer entsprechenden Warnung hin. Abbildung 3.5 zeigt die Vorgehensweise: In der Spalte *Filter* gibt Max Mustermann *negotiate* ein, um die Anzahl der angezeigten Optionen auf ein überschaubares Maß zu reduzieren. Wenn er dann auf *network.negotiate-auth.trusted-uris* doppelklickt, öffnet sich ein neues Fenster, in dem er dann die gewünschte URL eintragen kann. Hier ist das *https://www.example.com*.

Nachdem Herr Mustermann diese Änderung vorgenommen hat, kann er auf die Site *Geschützter Bereich* zugreifen. Das Ergebnis sehen Sie in Abbildung 3.6.

[2]Tatsächlich ist die Standardeinstellung bei Firefox auf einigen aktuellen Linux-Distributionen, *network.negotiate-auth.trusted-uris* auf den Wert *https://* zu setzen und damit gegenüber sämtlichen Sites Kerberos-Authentisierung zuzulassen, solange die Verbindung kein unverschlüsseltes HTTP verwendet.

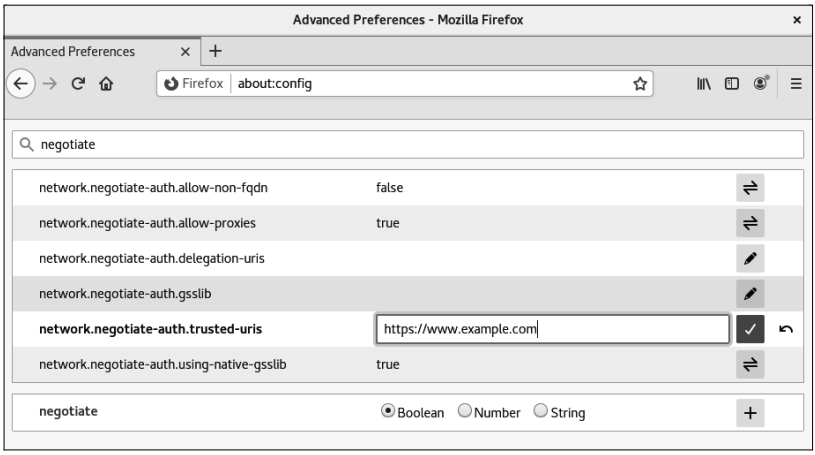


Abb. 3.5 *Kerberos-Konfiguration des Firefox-Webbrowsers*

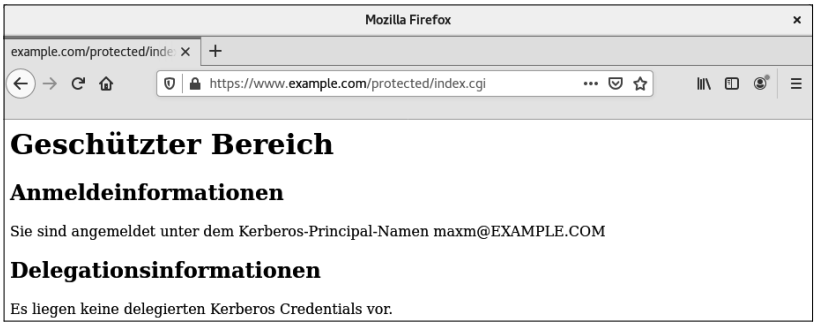


Abb. 3.6 *Die Kerberos-Authentisierung hat geklappt: Der Apache erkennt die Identität des Clients,* `maxm@EXAMPLE.COM`.

Das CGI-Skript zeigt unter *Anmeldeinformationen* die Identität des Clients an: `maxm@EXAMPLE.COM`. Das ist das Ergebnis der Kerberos-Authentisierung. Der Apache-Server und die Beispielanwendung können also sicher davon ausgehen, dass der HTTP-Zugriff auf den geschützten Bereich wirklich durch Max Mustermann erfolgt. Nach dem erfolgreichen Zugriff auf die geschützte Seite hat sich dessen Credential Cache wiederum erweitert: Listing 3.8 zeigt, dass nun auch ein HTTP-Ticket vorliegt.

Listing 3.8 *Credential Cache von Max Mustermann nach kerberisiertem Zugriff auf die geschützte Seite*

```
maxm@lx01:~$ klist
Ticket cache: FILE:/tmp/krb5cc_10000_zGOviZ
Default principal: maxm@EXAMPLE.COM

Valid starting      Expires             Service principal
06/22/2021 17:32:59 06/23/2021 03:32:59  krbtgt/EXAMPLE.
➥  COM@EXAMPLE.COM
       renew until 06/23/2021 17:32:57
06/22/2021 17:34:07 06/23/2021 03:32:59  ldap/kdc01.
➥  example.com@EXAMPLE.COM
       renew until 06/23/2021 17:32:57
```

```
06/22/2021 17:35:53 06/23/2021 03:32:59  host/lx02.example ⬅
➥  .com@EXAMPLE.COM
       renew until 06/23/2021 17:32:57
06/22/2021 22:29:20 06/23/2021 03:32:59  HTTP/lx02.example ⬅
➥  .com@EXAMPLE.COM
       renew until 06/23/2021 17:32:57
```

Interessant dabei ist, dass das Ticket den Namen `HTTP/lx02.example.com@EXAMPLE.COM` enthält, obwohl Herr Mustermann im Browser den DNS-Aliasnamen `www.example.com` angegeben hat. Das liegt daran, dass der Browser eine *Kanonisierung* des Hostnamens durchführt und anstelle des Aliasnamens den primären DNS-Namen verwendet (siehe auch Abschnitt 6.8.2).

Das CGI-Skript der Beispielumgebung kann auch Informationen zu delegierten Kerberos Tickets anzeigen. Intern ruft es dazu das `klist`-Kommando auf und stellt dessen Ausgabe dar, sofern delegierte Tickets vorhanden sind. Das ist in Abbildung 3.6 nicht der Fall, denn Max Mustermann hat keine Tickets weitergeleitet.

Um das zu tun, muss er noch einen weiteren Firefox-Konfigurationsparameter anpassen: *network.negotiate-auth.delegation-uris* listet die URLs derjenigen Sites, an die Tickets weitergeleitet werden sollen. Abbildung 3.7 zeigt, wie das im Falle von *https://www.example.com* aussieht.

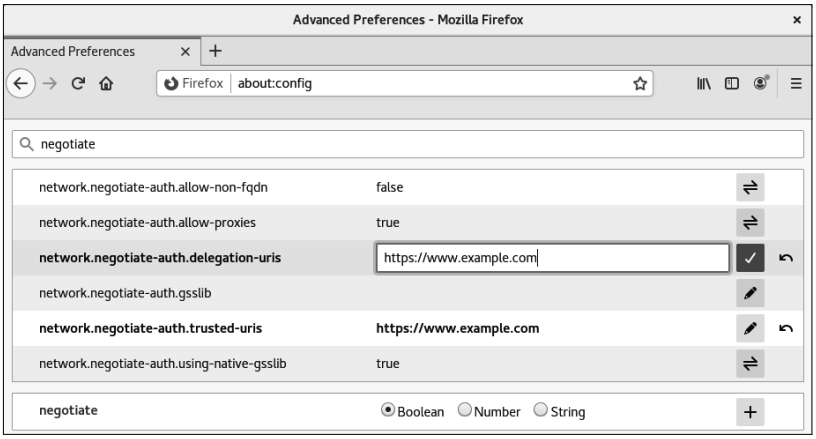


Abb. 3.7 *Delegation konfigurieren*

Bei einem erneuten Zugriff auf den geschützten Bereich führt das CGI-Skript der Beispielumgebung einige weitere Schritte durch (siehe Abbildung 3.8). Zuerst erkennt es, dass ein Credential Cache mit delegierten Tickets vorliegt, und zeigt diesen an: Der Credential Cache liegt unter `/tmp/krb5cc_apache_C0aEfV` und enthält ein TGT für den Client

maxm@EXAMPLE.COM. Auch hier fällt auf, dass bei diesem TGT das f-Flag (forwarded) gesetzt ist.

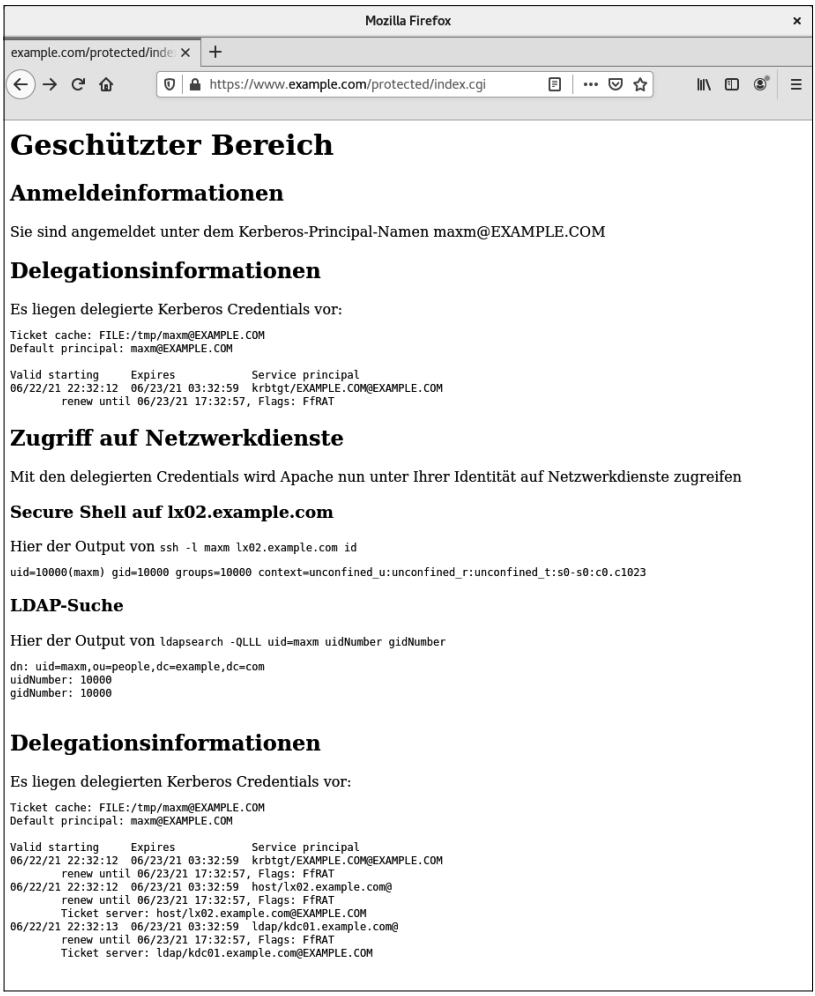


Abb. 3.8
Der Apache hat jetzt ein weitergeleitetes TGT von Max Mustermann. Damit kann er stellvertretend auf Netzwerkdienste zugreifen.

Das CGI-Skript benutzt das TGT nun analog zu Abschnitt 3.4 für Zugriffe auf die beiden Beispieldienste (Secure Shell auf lx02 und LDAP auf kdc01):

- Es führt den Befehl ssh -l maxm lx02.example.com id aus. Dadurch startet es eine SSH-Sitzung auf der lx02 unter der Identität von Max Mustermann (maxm). Innerhalb dieser SSH-Sitzung führt es das Kommando id aus, das einige Linux-Benutzereigenschaften zu maxm anzeigt.

- Daraufhin startet das CGI-Skript eine LDAP-Anfrage analog zu Abschnitt 3.4.

In beiden Fällen stellt das CGI-Skript die Ausgabe der Kommandos dar. Anschließend zeigt es noch den veränderten Credential Cache nach den Zugriffen an. Wie in Abschnitt 3.4 liegen auch hier nun ein Host Ticket und ein LDAP-Ticket vor.

Fazit

Der Apache kann mithilfe des weitergeleiteten TGT so auf Netzwerkdienste zugreifen, wie Max Mustermann das selbst könnte.

Achtung

Auch hier stellt das Ticket Forwarding einen sicherheitsrelevanten Vorgang dar. Nehmen Sie daher immer nur vertrauenswürdige Sites in *network.negotiate-auth.delegation-uris* auf. Darüber hinaus kann die beschriebene Vorgehensweise der Delegation speziell beim HTTP-Protokoll zu sehr großen Performance-Problemen führen (siehe Abschnitt 24.3).

3.7 Umgang mit dem Credential Cache

Herr Mustermann hat den Credential Cache bisher nur mit `klist` beobachtet. Es gibt aber noch andere Kerberos-Kommandos, mit denen man den Credential Cache verwalten kann:

- `kdestroy`
- `kinit`
- `kvno` und `kgetcred`

Diese Kommandos sind auch Thema von Kapitel 10 und 14, sollen hier aber bereits kurz eingeführt werden.

`kdestroy`

Das Kommando `kdestroy` dient dem sicheren Löschen eines Credential Cache. In Listing 3.9 führt `maxm` das Kommando `kdestroy` aus. Danach findet `klist` keine Tickets mehr vor.

```
maxm@lx01:~$ kdestroy
maxm@lx01:~$ klist
klist: No credentials cache found (ticket cache FILE:/tmp/ ➥
➥  krb5cc_10000_zGOviZ)
maxm@lx01:~$
```

Listing 3.9 *Tickets mit* `kdestroy` *sicher löschen*

`kinit`

In der Beispielumgebung könnte sich `maxm` nun ab- und wieder anmelden, um dabei ein neues TGT zu bekommen. Schneller geht es mit dem `kinit`-Kommando, das in Listing 3.10 dargestellt ist.

Listing 3.10 *Neues Ticket-Granting Ticket holen mit* `kinit`. *Danach zeigt* `klist` *wieder einen Credential Cache an.*

```
maxm@lx01:~$ kinit maxm@EXAMPLE.COM
Password for maxm@EXAMPLE.COM: P@ssw0rd
maxm@lx01:~$ klist
Ticket cache: FILE:/tmp/krb5cc_10000_zGOviZ
Default principal: maxm@EXAMPLE.COM
Valid starting      Expires             Service principal
08/21/2021 14:42:14 08/22/2021 00:42:14  krbtgt/EXAMPLE.
➥  COM@EXAMPLE.COM
       renew until 08/22/2021 14:42:11
maxm@lx01:~$
```

In Abschnitt 3.4 hat Herr Mustermann automatisch neue Service Tickets erhalten, wenn er auf den jeweiligen Dienst zugegriffen hat. Man kann sich Tickets aber auch direkt holen, ohne ein spezielles Clientkommando (wie `ssh` oder `ldapsearch`) zu benutzen: MIT Kerberos enthält das

`kvno`

Kommando `kvno`, das ein Ticket für einen speziellen Netzwerkdienst holt (auch wenn das nicht der eigentliche Zweck von `kvno` ist). Heimdal kennt dazu den Befehl `kgetcred`. Listing 3.11 stellt diesen Vorgang dar.

Listing 3.11 *Service Tickets mit* `kvno` *beziehen*

```
maxm@lx01:~$ kvno host/lx02.example.com@EXAMPLE.COM
host/lx02.example.com@EXAMPLE.COM: kvno = 2
maxm@lx01:~$ klist
Ticket cache: FILE:/tmp/krb5cc_10000_zGOviZ
Default principal: maxm@EXAMPLE.COM
Valid starting      Expires             Service principal
08/21/2021 14:42:14 08/22/2021 00:42:14  krbtgt/EXAMPLE.
➥  COM@EXAMPLE.COM
       renew until 08/22/2021 14:42:11
08/21/2021 14:43:44 08/22/2021 00:42:14  host/lx02.example
➥  .com@EXAMPLE.COM
       renew until 08/22/2021 14:42:11
maxm@lx01:~$
```

3.8 Zusammenfassung

In einer kerberisierten Umgebung erhalten Anwender:innen bei der lokalen Anmeldung ein *Ticket-Granting Ticket (TGT)*. Das TGT und auch alle weiteren *Service Tickets* liegen im *Credential Cache*.

Kerberos-Unterstützung muss bei allen kerberisierten Programmen softwareseitig vorhanden sein. Darüber hinaus ist auch eine Konfiguration der Services und Clients für die Verwendung von Kerberos nötig, Sie haben das am Beispiel des Firefox-Webbrowsers gesehen.

Netzwerk-Clientprogramme holen bei Bedarf die Tickets für ihre Services. Daher füllt sich der Credential Cache bei jedem Zugriff auf einen neuen Netzwerkdienst.

Sie haben auch erfahren, dass Kerberos Tickets *Flags* wie *Forwardable* oder *Forwarded* besitzen. Mehr zu diesen Flags werden Sie in Abschnitt 6.2 erfahren.

Durch das Weiterleiten des TGT *(Ticket Forwarding)* können Clients ihre Identität einem Dienst zur Verfügung stellen. Dieser kann dann die Clientidentität annehmen, wenn er auf andere Dienste zugreift. Ticket Forwarding stellt somit eine Variante der in Abschnitt 2.3 beschriebenen *Delegation* dar.

Anwender können ihren Credential Cache mit Tools wie `klist`, `kinit`, `kvno` und `kdestroy` anzeigen und verwalten.