5.1 Lektion 1

Zertifikat:	Linux Essentials
Version:	1.6
Thema:	5 Sicherheit und Dateiberechtigungen
Lernziel:	5.1 Sicherheitsgrundlagen und Identifizierung von Benutzertypen
Lektion:	1 von 1

Einführung

Diese Lektion befasst sich mit den **Grundlagen von Benutzerkonten**, Zugriffskontrollen, der **Sicherheit lokaler Linux-Systeme**, den Command Line Interface (CLI)-Tools in einem Linux-System für grundlegende Sicherheitszugriffskontrollen sowie den Hauptdateien zur Unterstützung von Benutzer- und Gruppenkonten, einschließlich derjenigen, die die Zuweisung von Zugriffsrechten steuern.

Die Sicherheit in Linux-Systemen **basiert auf Unix-Zugriffskontrollen**, die zwar fast fünfzig Jahre alt, aber im Vergleich zu einigen deutlich jüngeren Consumer-Betriebssystemen sehr effektiv sind. Selbst einige beliebte Unix-basierte Betriebssysteme neigen dazu, sich “Freiheiten” im Hinblick auf den “einfachen Zugang” zu nehmen, was Linux vermeidet.

Moderne Linux-Desktop-Umgebungen und -Schnittstellen vereinfachen Erstellung und Verwaltung von Benutzern und automatisieren oft die Zuordnung von Zugriffskontrollen, wenn sich ein Benutzer anmeldet — z.B. am Bildschirm, für das Audio-System und andere Dienste — und erfordern praktisch keinen manuellen Eingriff des Systemadministrators. Es ist jedoch wichtig, die zugrundeliegenden Konzepte des Linux-Betriebssystems zu verstehen.

Benutzerkonten

Sicherheit umfasst viele Konzepte, und eines der verbreitetsten ist das allgemeine Konzept der Zugangskontrolle. Bevor man sich mit **Dateizugriffskontrollen** wie **Eigentum und Berechtigungen befasst**, muss man die Grundkonzepte der Benutzerkonten (*Accounts*) unter Linux verstehen, die in verschiedene Arten unterteilt sind.

Jeder Benutzer auf einem Linux-System hat ein ihm zugeordnetes Konto (Account), das neben den Login-Informationen (wie **Benutzername und Passwort)** auch bestimmt, wie und wo der Benutzer mit dem System interagieren kann. Berechtigungen und Zugriffskontrollen definieren die “Grenzen”, innerhalb derer jeder Benutzer agiert.

Identifier (UIDs/GIDs)

Die ***User* und *Group Identifiers* (UIDs/GIDs)** sind die grundlegenden, durch-nummerierten Referenzen auf Accounts. Frühe Implementierungen waren auf 16-Bit-Ganzzahlen (Werte 0 bis 65535) beschränkt, aber die Systeme des 21. Jahrhunderts

unterstützen 64-Bit-UIDs und -GIDs. Benutzer und Gruppen werden unabhängig voneinander gezählt, so dass dieselbe ID sowohl für einen Benutzer als auch für eine Gruppe stehen kann.

Jeder Benutzer hat nicht nur eine UID, sondern auch eine ***primäre GID***. Die primäre GID für einen Benutzer kann für diesen Benutzer einmalig sein und von **keinem anderen Benutzer** verwendet werden. Aber diese Gruppe kann auch zahlreiche Benutzer umfassen. Neben der primären Gruppe kann jeder Benutzer auch Mitglied anderer Gruppen sein.

Standardmäßig ist unter **Linux jeder Benutzer einer Gruppe mit demselben Namen** wie sein Benutzername und derselben GID wie seine UID zugeordnet. Erstellen Sie zum Beispiel einen **neuen Benutzer** mit dem **Namen** `newuser`, so ist per Default auch seine **Standardgruppe** `newuser`.

Der Superuser-Account

Unter Linux hat das Konto des **Superusers** `root` **immer UID 0**. Der Superuser wird manchmal als "**Systemadministrator**" bezeichnet und hat unbegrenzten Zugriff auf und Kontrolle über das System einschließlich seiner Benutzer.

Die **Standardgruppe** für den Superuser hat die GID 0 und heißt **ebenfalls** `root`. Das Heimatverzeichnis des Superusers ist ein dediziertes Top-Level-Verzeichnis `/root`, das ausschließlich dem Benutzer `root` selbst zugänglich ist.

Standard-Benutzerkonten

Alle Accounts außer `root` sind technisch betrachtet reguläre Benutzerkonten, aber auf einem Linux-System bezeichnet der Begriff *Benutzerkonto* oft ein **"reguläres" (unprivilegiertes) Benutzerkonto.** Diese haben (abgesehen von einigen Ausnahmen) typischerweise die folgenden Eigenschaften:

- **Die UIDs starten bei 1000**, sind also 4-stellig (bei einigen älteren Systemen auch bei 500).
- Es gibt ein **definiertes Home-Verzeichnis**, in der Regel ein Unterverzeichnis von `/home`, abhängig von der lokalen Systemkonfiguration.
- Es gibt eine **definierte Login-Shell**, unter Linux ist die Standard-Shell die *Bourne Again Shell* (`/bin/bash`), obwohl auch andere möglich sind.

Ist einem Benutzerkonto keine gültige Shell in seinen Attributen zugewiesen, kann der Benutzer keine interaktive Shell öffnen. Üblicherweise wird `/sbin/nologin` als ungültige Shell gesetzt. Das ist etwa dann sinnvoll, wenn der Benutzer nur für andere Dienste als Konsolen- oder SSH-Zugriff authentifiziert wird, z.B. Secure FTP (`sftp`).

Note Um Verwirrung zu vermeiden, bezeichnen die Begriffe *Benutzerkonto* bzw. *User Account* im Folgenden ausschließlich reguläre Benutzerkonten. Im Gegensatz dazu stehen *Systemkonto* bzw. *System Account* für ein Linux-Benutzerkonto vom Typ System-benutzerkonto bzw. System User Account.

System Accounts

System Accounts werden **typischerweise** bereits bei der **Systeminstallation erstellt**, und zwar für Programme und Dienste, die nicht als Superuser laufen. Im Idealfall wäre dies alles bestriebssystemspezifisch.

Die Systemkonten variieren, aber sie haben folgende Eigenschaften:

- **UIDs** liegen in der Regel unter 100, sind also 2-stellig, oder zwischen 500 und 1000.
- Sie haben entweder ***kein* dediziertes Home-Verzeichnis** oder ein Verzeichnis, das normalerweise nicht unter `/home` liegt.
- Sie haben **keine gültige Login-Shell** (typischerweise `/sbin/nologin`), mit seltenen Ausnahmen.

Die meisten Systemkonten unter Linux **benötigen nie eine Anmeldung** und folglich keine definierte Shell in ihren Attributen. Viele Prozesse, die System Accounts gehören und von diesen ausgeführt werden, zweigt das Systemmanagement in eine eigene Umgebung ab, die mit dem angegebenen System Account läuft. In der Regel haben diese Accounts eingeschränkte oder meist sogar *keine* Berechtigungen.

Note Aus Sicht der Prüfung Linux Essentials sind System Accounts solche mit UID <1000 mit 2 oder 3-stelligen UIDs (und GIDs).

Im Allgemeinen sollten System Accounts *keine* gültige Login-Shell haben, da diese in den meisten Fällen ein Sicherheitsrisiko wäre.

Service Accounts

Service Accounts werden typischerweise bei der Installation und Konfiguration von Diensten erstellt, ähnlich wie Systemkonten für Programme und Dienste, die nicht als Superuser ausgeführt werden.

In vielen Dokumentationen werden System und Service Accounts nicht unterschieden und die Begriffe synonym gebraucht. Bei beiden liegen die Home-Verzeichnisse typischerweise außerhalb von `/home`, wenn sie überhaupt definiert sind (Service Accounts haben eher eines als System Accounts), und beide haben keine gültige Login-Shell. Obwohl es keine verbindliche Definition gibt, liegt der Hauptunterschied zwischen System- und Servicekonten in UID und GID:

System Account
: UID/GID <100 (2-stellig) oder <500-1000

Service Account
: UID/GID >1000 (4-und-mehr-stellig), aber kein “Standard” or “regulärer” Benutzer-Account.

Einige Linux-Distributionen haben noch reservierte Servicekonten mit UID <100, die man auch als Systemkonto betrachten könnte, obwohl sie nicht bei der Systeminstallation erstellt werden. So hat der Benutzer für den Apache-Webserver bei Fedora-basierten Linux-Distributionen (einschließlich Red Hat) die UID (und GID) 48 und ist damit eindeutig ein Systemkonto, obwohl er ein Home-Verzeichnis hat (normalerweise unter `/usr/share/httpd` oder `/var/www/html/`).

Note Aus Sicht der Prüfung Linux Essentials sind Systemkonten jene mit UID <1000, während reguläre Benutzerkonten UID >1000 haben. Da die regulären Benutzerkonten >1000 sind, können diese UIDs auch Servicekonten enthalten.

Login-Shells und Home-Verzeichnisse

Einige Konten verfügen über eine Login-Shell, während andere aus Sicherheitsgründen keine haben, da sie keinen interaktiven Zugriff benötigen. Die

Standard-Login-Shell in den meisten Linux-Distributionen ist die *Bourne Again Shell* (`bash`), aber es können auch andere Shells verfügbar sein, wie die C-Shell (`csh`), die Korn-Shell (`ksh`) oder die Z-Shell (`zsh`), um nur einige zu nennen.

Ein Benutzer kann seine Login-Shell mit dem Befehl `chsh` ändern. Standardmäßig läuft der Befehl im interaktiven Modus und zeigt eine Eingabeaufforderung, die nach der gewünschten Shell fragt. Die Antwort sollte der vollständige Pfad zum Binary der Shell sein:

```
$ chsh

Changing the login shell for emma
Enter the new value, or press ENTER for the default
        Login Shell [/bin/bash]: /usr/bin/zsh
```

Sie können den Befehl auch im nicht-interaktiven Modus ausführen, wobei auf den Parameter `-s` der Pfad zum Binary folgt:

```
$ chsh -s /usr/bin/zsh
```

Die meisten Accounts haben ein definiertes Home-Verzeichnis. Unter Linux ist dies in der Regel der einzige Ort, wo dieser Benutzer sicher Schreibrechte hat — mit einigen Ausnahmen, z.B. temporäre Dateisystembereiche. Einige Accounts sind jedoch bewusst so eingerichtet, dass sie aus Sicherheitsgründen keinen Schreibzugriff auf ihr eigenes Home-Verzeichnis haben.

Benutzerinformationen abfragen

Das Auflisten grundlegender Benutzerinformationen ist gängige Praxis auf einem Linux-System. In einigen Fällen müssen Benutzer den Benutzer wechseln und die Berechtigung erweitern, um privilegierte Aufgaben auszuführen.

Selbst Benutzer haben die Möglichkeit, Attribute und Zugriff über die Befehlszeile mit den folgenden Befehlen aufzulisten. Grundlegende Informationen in einem begrenzten Kontext sind keine privilegierte Operation.

Die Anzeige aktueller Informationen zu einem Benutzer auf der Befehlszeile ist ein simpler Befehl: `id`. Die Ausgabe variiert je nach Login-ID:

```
$ id
uid=1024(emma) gid=1024(emma)
1024(emma),20(games),groups=10240(netusers),20480(netadmin)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Die Ausgabe für den Benutzer (`emma`) zeigt folgende Merkmale:

- `1024` ist die User ID (UID), gefolgt vom Benutzernamen (Login-Name) in Klammern.
- `1024` ist die *primäre* Gruppen-ID (GID), gefolgt vom Gruppennamen in Klammern.
- Eine Liste der zusätzlichen GIDs (und Gruppennamen), denen der Benutzer ebenfalls angehört.

Die Anzeige der letzten Anmeldungen von Benutzern am System erfolgt mit dem Befehl `last`:

```
$ last
emma     pts/3        ::1              Fri Jun 14 04:28   still logged in
reboot   system boot  5.0.17-300.fc30. Fri Jun 14 04:03   still running
```

```
reboot   system boot  5.0.17-300.fc30. Wed Jun  5 14:32 - 15:19  (00:46)
reboot   system boot  5.0.17-300.fc30. Sat May 25 18:27 - 19:11  (00:43)
reboot   system boot  5.0.16-100.fc28. Sat May 25 16:44 - 17:06  (00:21)
reboot   system boot  5.0.9-100.fc28.x Sun May 12 14:32 - 14:46  (00:14)
root     tty2                         Fri May 10 21:55 - 21:55  (00:00)
        ...
```

Die in den Spalten aufgeführten Informationen können variieren, aber nennenswerte Einträge in diesem Beispiel sind:

- Ein Benutzer (`emma`) hat sich über das Netzwerk angemeldet (Pseudo TTY `pts/3`) und ist immer noch angemeldet.
- Zeitpunkt des aktuellen Bootvorgangs und der Kernel; hier ca. 25 Minuten vor dem Login des Benutzers.
- Der Superuser (`root`) war Mitte Mai über eine virtuelle Konsole (TTY `tty2`) kurz angemeldet.

Eine Variante von `last` ist der Befehl `lastb`, der die letzten fehlgeschlagenen Anmeldeversuche auflistet.

Die Befehle `who` und `w` listen nur die aktiven Anmeldungen am System auf:

```
$ who
emma  pts/3        2019-06-14 04:28 (::1)

$ w
 05:43:41 up  1:40,  1 user,  load average: 0.25, 0.53, 0.51
USER     TTY        LOGIN@   IDLE   JCPU   PCPU WHAT
emma  pts/3     04:28    1:14m  0.04s  0.04s -bash
```

Beide Befehle zeigen zum Teil die gleichen Informationen. So ist zum Beispiel ein Benutzer (`emma`) über ein Pseudo-TTY-Device (`pts/3`) angemeldet und der Zeitpunkt der Anmeldung ist 04:28.

Der Befehl `w` listet weitere Informationen auf, einschließlich der folgenden:

- die aktuelle Zeit und wie lange das System hochgefahren ist
- wie viele Benutzer verbunden sind
- die durchschnittliche Systemlast (`load average`) der letzten 1, 5 und 15 Minuten

Zudem werden weitere Informationen für jede aktive Benutzersitzung angezeigt:

- Auswahl der gesamten CPU-Nutzungszeiten (`IDLE`, `JCPU` und `PCPU`)
- Aktueller Prozess (`-bash`) — die gesamte CPU-Nutzung dieses Prozesses ist der letzte Eintrag (`PCPU`).

Beide Befehle haben weitere Optionen, um zusätzliche Informationen aufzulisten.

Benutzerwechsel und Berechtigungen erweitern

In einer idealen Welt müssten Benutzer ihre Dateirechte nicht erweitern, um ihre Aufgaben zu erledigen: Das System würde stets “einfach funktionieren”, und alles wäre für verschiedene Zugriffe konfiguriert.

Glücklicherweise funktioniert Linux out-of-the-box tatsächlich so für die meisten Benutzer, die keine Systemadministratoren sind, obwohl es stets dem *Least Privilege*-Sicherheitsmodell folgt, also nicht mehr Rechte gewährt als unbedingt notwendig.

Es gibt jedoch Befehle, die eine Rechteerweiterung bei Bedarf erlauben. Zwei der wichtigsten sind `su` und `sudo`.

Auf den meisten Linux-Systemen dient der Befehl `su` nur der Vergabe von root-Rechten an den aktuellen Benutzer, sofern nach dem Befehlsnamen kein Benutzername angegeben wird. Obwohl der Befehl auch verwendet werden kann, um zu einem anderen Benutzer zu wechseln, ist das keine gute Praxis: Benutzer sollten sich von einem anderen System, über das Netzwerk oder die physische Konsole bzw. das Terminal auf dem System anmelden.

```
emma ~$ su -
Password:
root ~#
```

Nach der Eingabe des Superuser- (`root`-)Passwortes hat der Benutzer eine Superuser-Shell (beachten Sie das `#` am Ende der Eingabeaufforderung) und kann damit als Superuser (`root`) agieren.

Die Weitergabe von Passwörtern ist sehr schlechte Sicherheitspraxis, so dass es in einem modernen Linux-System nur sehr wenig (wenn überhaupt) Bedarf für den `su`-Befehl geben sollte.

Das Dollar-Symbol (`$`) sollte den Kommandozeilen-Prompt für eine nicht privilegierte Benutzer-Shell abschließen, während das Hash-Symbol (`#`) den Kommandozeilen-Prompt für den Superuser (`root`) beenden sollte. Es wird dringend empfohlen, das letzte Zeichen eines Prompts *nie* von diesem allgemein gültigen Standard abweichend zu definieren, da diese Nomenklatur auch in Lernmaterialien, einschließlich dieser, verwendet wird.

Warning

Wechseln Sie niemals zum Superuser (`root`), ohne den Parameter der Login-Shell (`-`) zu übergeben. Sofern nicht ausdrücklich vom Betriebssystem oder Softwarehersteller anders angewiesen, führen Sie `su` immer `su -` aus, mit extrem begrenzten Ausnahmen. Benutzerumgebungen können unerwünschte Konfigurationsänderungen und Probleme verursachen, wenn sie im Vollberechtigungsmodus als Superuser benutzt werden.

Was ist das größte Problem bei der Verwendung von `su`, um zum Superuser (`root`) zu wechseln? Wenn die Sitzung eines regulären Benutzers kompromittiert wurde, könnte das Passwort des Superusers (`root`) abgefangen werden. Hier kommt “Switch User Do” (oder “Superuser Do”) ins Spiel:

```
$ cat /sys/devices/virtual/dmi/id/board_serial
cat: /sys/devices/virtual/dmi/id/board_serial: Permission denied

$ sudo cat /sys/devices/virtual/dmi/id/board_serial
[sudo] password for emma:
/6789ABC/
```

Im vorangegangenen Beispiel versucht der Benutzer, die Seriennummer seiner Systemplatine auszulesen. Der Zugriff wird jedoch verweigert, da diese Information als privilegiert gekennzeichnet ist.

Indem er jedoch `sudo` nutzt und sein eigenes Passwort eingibt, authentifiziert der Benutzer, wer er ist. Wenn er in der `sudoers`-Konfiguration autorisiert wurde, diesen Befehl mit den erlaubten Optionen auszuführen, wird es funktionieren.

Tip Standardmäßig wird der erste autorisierte `sudo`-Befehl nachfolgende `sudo`-Befehle für eine (sehr kurze) Zeitspanne authentifizieren. Dies ist vom Systemadministrator konfigurierbar.

Access-Control-Dateien

Fast alle Betriebssysteme speichern Zugriffskontrollen an bestimmten Orten. Unter Linux sind dies typischerweise Textdateien im Verzeichnis `/etc`, in dem üblicherweise die Systemkonfigurationsdateien liegen. Standardmäßig ist dieses Verzeichnis für jeden Benutzer des Systems lesbar, aber nur durch `root` beschreibbar.

Die wichtigsten Dateien in Bezug auf Benutzerkonten, Attribute und Zugriffskontrolle sind:

`/etc/passwd`
: Diese Datei enthält grundlegende Informationen über die Benutzer auf dem System, einschließlich UID und GID, Home-Verzeichnis, Shell usw. Trotz des Namens werden hier keine Passwörter gespeichert.

`/etc/group`
: Diese Datei enthält grundlegende Informationen über alle Benutzergruppen auf dem System, wie Gruppenname, GID und Mitglieder.

`/etc/shadow`
: Hier werden die Benutzerpasswörter gespeichert, die aus Sicherheitsgründen gehasht sind.

`/etc/gshadow`
: Diese Datei enthält detailliertere Informationen über Gruppen, einschließlich eines gehashten Passworts, mit dem Benutzer vorübergehend Mitglied der Gruppe werden können, einer Liste von Benutzern, die zu einem bestimmten Zeitpunkt Mitglied der Gruppe werden können, und einer Liste von Gruppenadministratoren.

Warning Diese Dateien sollten niemals direkt bearbeitet werden. In dieser Lektion geht es nur um die Informationen, die in diesen Dateien gespeichert sind, nicht um deren Bearbeitung.

Standardmäßig kann jeder Benutzer in `/etc` wechseln und die Dateien `/etc/passwd` und `/etc/group` lesen. Und ebenso standardmäßig darf kein Benutzer außer `root` die Dateien `/etc/shadow` oder `/etc/gshadow` lesen.

Es gibt zudem Dateien, die mit der grundlegenden Rechteerweiterung auf Linux-Systemen zu tun haben, etwa über die Befehle `su` und `sudo`. Standardmäßig sind diese nur für den Benutzer `root` zugänglich:

`/etc/sudoers`
: Diese Datei kontrolliert, wer den `sudo`-Befehl ausführen darf und wie.

`/etc/sudoers.d`
: Dieses Verzeichnis kann Dateien enthalten, die die Einstellungen der Datei `sudoers` ergänzen.

Für die Prüfung Linux Essentials muss man nur den Pfad und den Dateinamen der Standard `sudo`-Konfigurationsdatei `/etc/sudoers` kennen. Ihre Konfiguration geht über den Rahmen dieser Materialien hinaus.

Warning Auch wenn `/etc/sudoers` eine Textdatei ist, sollte sie nie direkt bearbeitet werden. Sind Änderungen an ihrem Inhalt notwendig, sollten sie mit dem Werkzeug `visudo` vorgenommen werden.

Die Datei `/etc/passwd`

Die Datei `/etc/passwd` wird allgemein als “Passwortdatei” bezeichnet. Jede Zeile enthält mehrere Felder, die immer durch einen Doppelpunkt (`:`) abgegrenzt sind. Trotz des Namens wird der eigentliche Einweg-Passwort-Hash heute nicht mehr in dieser Datei gespeichert.

Die typische Syntax für eine Zeile in dieser Datei lautet:

```
USERNAME:PASSWORD:UID:GID:GECOS:HOMEDIR:SHELL
```

Die Felder haben folgende Bedeutung:

`USERNAME`
: Der Benutzername (auch Login-Name genannt), wie `root`, `nobody`, `emma`.

`PASSWORD`
: Ursprünglich der Ort für den Passwort-Hash. Heute fast immer `x`, was anzeigt, dass das Passwort in der Datei `/etc/shadow` gespeichert ist.

`UID`
: Benutzer-ID (UID), wie `0`, `99`, `1024`.

`GID`
: Standard-Gruppen-ID (GID), wie `0`, `99`, `1024`.

`GECOS`
: Eine CSV-Liste mit Benutzerinformationen, einschließlich Name, Standort, Telefonnummer, zum Beispiel: `Emma Smith,42 Douglas St,555.555.5555`

`HOMEDIR`
: Pfad zum Heimatverzeichnis des Benutzers, wie `/root`, `/home/emma` etc.

`SHELL`
: Die Standard-Shell für diesen Benutzer, wie `/bin/bash`, `/sbin/nologin`, `/bin/ksh` etc.

Zum Beispiel beschreibt die folgende Zeile den Benutzer `emma`:

```
emma:x:1000:1000:Emma Smith,42 Douglas St,555.555.5555:/home/emma:/bin/bash
```

Das GECOS-Feld

Das GECOS-Feld enthält drei oder mehr Felder, die durch Komma (`,`) abgegrenzt sind — es handelt sich also um eine Liste *Comma Separated Values* (CSV). Obwohl

es keinen definierten Standard gibt, sind die Felder normalerweise in der folgenden Reihenfolge angeordnet:

```
NAME,LOCATION,CONTACT
```

Die Felder haben folgende Bedeutung:

`NAME`
: Der vollständige Name des Benutzers oder der Name der Software im Falle eines Servicekontos.

`LOCATION`
: Normalerweise der physische Standort des Benutzers innerhalb eines Gebäudes, die Zimmernummer oder die Kontaktabteilung oder -person im Falle eines Servicekontos.

`CONTACT`
: Listet Kontaktinformationen wie die Telefonnummer zu Hause oder am Arbeitsplatz auf.

Zusätzliche Felder können zusätzliche Kontaktinformationen enthalten, wie z.B. eine Privatnummer oder E-Mail-Adresse. Um die Informationen im GECOS-Feld zu ändern, verwenden Sie den Befehl `chfn` und beantworten die Fragen, wie unten gezeigt. Wenn nach dem Befehlsnamen kein Benutzername angegeben wird, ändern Sie die Informationen für den aktuellen Benutzer:

```
$ chfn

Changing the user information for emma
Enter the new value, or press ENTER for the default
        Full Name: Emma Smith
        Room Number []: 42
        Work Phone []: 555.555.5555
        Home Phone []: 555.555.6666
```

Die Datei `/etc/group`

Die Datei `/etc/group` enthält durch einen Doppelpunkt (`:`) abgegrenzte Felder mit grundlegenden Informationen über die Gruppen auf dem System. Sie wird häufig auch als “Group File” oder “Gruppendatei” bezeichnet. Die Syntax für jede Zeile lautet:

```
NAME:PASSWORD:GID:MEMBERS
```

Die Felder haben folgende Bedeutung:

`NAME`
: Gruppenname, wie `root`, `users`, `emma` etc.

`PASSWORD`
: Ursprünglicher Ort eines optionalen Gruppenpasswort-Hashes. Heute fast immer `x`, was anzeigt, dass das Passwort (falls definiert) in der Datei `/etc/gshadow` gespeichert ist.

`GID`
: Gruppen-ID (GID), wie `0`, `99`, `1024`.

`MEMBERS`

Kommaseparierte Liste von Benutzernamen, die Mitglieder der Gruppe sind, wie `jsmith,emma`.

Das folgende Beispiel zeigt eine Zeile mit Informationen über die Gruppe `students`:

```
students:x:1023:jsmith,emma
```

Wenn die Gruppe die primäre Gruppe für einen Benutzer ist, muss der Benutzer nicht im Mitgliederfeld aufgeführt werden. Wenn ein Benutzer aufgeführt wird, ist er redundant — d.h. es gibt keine Änderung in der Funktionalität, ob aufgeführt oder nicht.

Note Die Verwendung von Passwörtern für Gruppen liegt außerhalb des Rahmens dieser Lektion, jedoch wird, falls definiert, der Passwort-Hash in der Datei `/etc/gshadow` gespeichert — ebenfalls nicht Thema dieses Abschnitts.

Die Datei `/etc/shadow`

Die folgende Tabelle listet die Attribute auf, die in der Datei `/etc/shadow`, allgemein als "Shadow-Datei" bezeichnet, gespeichert sind. Die Datei enthält Felder, die immer durch einen Doppelpunkt (`:`) begrenzt sind. Obwohl die Datei viele Felder hat, liegen die meisten außerhalb des Rahmens dieser Lektion, mit Ausnahme der ersten beiden.

Die Syntax für eine Zeile in dieser Datei lautet:

```
USERNAME:PASSWORD:LASTCHANGE:MINAGE:MAXAGE:WARN:INACTIVE:EXPDATE
```

Die Felder haben folgende Bedeutung:

`USERNAME`
: Der Benutzername (wie in `/etc/passwd`), wie `root`, `nobody`, `emma`.

`PASSWORD`
: Ein Einweg-Hash des Passworts, inklusive vorangestelltem Salt. Zum Beispiel: `!!`, `!$1$01234567$ABC…`, `$6$012345789ABCDEF$012…`.

`LASTCHANGE`
: Datum der letzten Passwortänderung in Tagen seit der "Epoche", z.B. `17909`.

`MINAGE`
: Mindestpasswortalter in Tagen.

`MAXAGE`
: Maximales Passwortalter in Tagen.

`WARN`
: Warnfrist vor Ablauf des Passworts in Tagen.

`INACTIVE`
: Maximales Passwortalter nach Ablauf der Gültigkeitsdauer in Tagen.

`EXPDATE`
: Datum des Ablaufs des Passworts in Tagen seit der "Epoche".

In dem Beispiel unten sehen Sie einen Beispieleintrag aus der Datei `/etc/shadow`. Beachten Sie, dass einige Werte wie `INACTIVE` und `EXPDATE` undefiniert sind.

```
emma:$6$nP532JDDogQYZF8I$bjFNh9eT1xpb9/n6pmjlIwgu7hGjH/eytSdttbmVv0MlyTMFgB
IXESFNUmTo9EGxxH1OT1HGQzR0so4n1npbE0:18064:0:99999:7:::
```

Die “Epoche” eines POSIX-Systems ist Mitternacht (0000), Universal Coordinate Time (UTC), Donnerstag, 1. Januar 1970. Die meisten POSIX-Daten und -Zeiten sind entweder in Sekunden seit “Epoche” oder, wie im Fall der Datei `/etc/shadow`, in Tagen seit “Epoche” angegeben.

Note Die Shadow-Datei ist so konzipiert, dass sie nur vom Superuser und ausgewählten Kernsystem-Authentifizierungsdiensten gelesen werden kann, die den Einweg-Passwort-Hash bei der Anmeldung oder einer anderen Authentifizierungsprozedur überprüfen.

Obwohl es verschiedene Authentifizierungslösungen gibt, ist die elementare Methode der Passwortspeicherung die einseitige *Hash-Funktion*. Dies geschieht, damit das Passwort niemals im Klartext auf einem System gespeichert wird, da die Hash-Funktion nicht umkehrbar ist. Sie können ein Passwort in einen Hash umwandeln, aber (im Idealfall) ist es nicht möglich, einen Hash wieder in ein Passwort umzuwandeln.

Es ist eine Brute-Force-Methode erforderlich, um alle Kombinationen eines Passworts zu hashen, bis eines übereinstimmt. Um die Wahrscheinlichkeit zu verringern, dass ein Passwort-Hash auf einem System gecrackt wird, verwenden Linux-Systeme ein zufälliges “Salz” (Salt) auf jedem Passwort-Hash für einen Benutzer, so dass der Hash für ein Benutzer-Passwort auf einem Linux-System in der Regel nicht dasselbe ist wie auf einem anderen Linux-System, auch wenn das Passwort dasselbe ist.

In der Datei `/etc/shadow` kann das Passwort verschiedene Formen haben, die typischerweise die folgenden sind:

`!!`
: Ein “deaktiviertes” Konto (keine Authentifizierung möglich), zu dem kein Passwort-Hash gespeichert ist.

`!$1$01234567$ABC…`
: Ein “deaktiviertes” Konto (am beginnenden Ausrufezeichen zu erkennen) mit einer vorherigen Hash-Funktion, Hash-Salt und Hash-String gespeichert.

`$1$0123456789ABC$012…`
: Ein “aktives” Konto mit Hash-Funktion, Hash-Salt und Hash-String gespeichert.

Die Hash-Funktion, das Hash-Salt und der Hash-String werden durch ein Dollar-Symbol (`$`) eingeleitet und abgegrenzt. Die Länge des Hash-Salzes muss 8-16 Zeichen betragen. Beispiele für die drei häufigsten sind:

`$1$01234567$ABC…`
: Eine Hash-Funktion über MD5 (`1`) mit einer Beispiel-Hash-Länge von 8.

`$5$01234567ABCD$012…`
: Eine Hash-Funktion über SHA256 (`5`) mit einer Beispiel-Hash-Länge von 12.

`$6$01234567ABCD$012…`
: Eine Hash-Funktion über SHA512 (`6`) mit einer Beispiel-Hash-Länge von 12.

Note Die MD5-Hash-Funktion gilt bei dem heutigen (ab 2010er) Niveau der ASIC- und sogar der allgemeinen Rechenleistung der SIMD als kryptographisch unsicher. So erlauben z.B. die US Federal Information Processing Standards (FIPS) die Verwendung von MD5 nicht für kryptographische Funktionen, nur sehr eingeschränkte Aspekte der Validierung, aber nicht die Integrität digitaler Signaturen oder ähnliche Zwecke.

Für die Prüfung Linux Essentials müssen Sie lediglich verstehen, dass der Passwort-Hash für einen lokalen Benutzer nur in der Datei `/etc/shadow` gespeichert wird, die nur Authentifizierungsdienste lesen können und die der Superuser über andere Kommandos modifizieren kann.

Geführte Übungen

1. Betrachten Sie die folgende Ausgabe des Befehls `id`:

```
$ id emma
uid=1000(emma) gid=1000(emma)
groups=1000(emma),4(adm),5(tty),10(uucp),20(dialout),27(sudo),46(plug
dev)
```

   - In welchen Dateien werden die folgenden Attribute gespeichert?

UID und GID	
Gruppen	

   - In welcher Datei wird zusätzlich das Benutzerpasswort gespeichert?

2. Welche der folgenden Arten von Kryptographie wird standardmäßig verwendet, um Passwörter lokal auf einem Linux-System zu speichern?
   - Asymmetrisch
   - Einweg-Hash
   - Symmetrisch
   - ROT13
3. Wenn ein Konto eine User-ID (UID) unter 1000 hat, um welche Art von Konto handelt es sich?
4. Wie erhalten Sie eine Liste der aktiven Logins in Ihrem System und deren Anzahl?
5. Der Befehl `grep` liefert das untenstehende Ergebnis mit Informationen über den Benutzer `emma`:

```
$ grep emma /etc/passwd
emma:x:1000:1000:Emma Smith,42 Douglas
St,555.555.5555,:/home/emma:/bin/ksh
```

   Füllen Sie die Tabelle mit den entsprechenden Informationen aus, indem Sie die Ausgabe des vorherigen Befehls nutzen:

Benutzername	
Passwort	
UID	
Primäre GID	
GECOS	
Home-Verzeichnis	
Shell	

Offene Übungen

1. Vergleichen Sie die Ergebnisse von `last` mit `w` und `who`. In welchen Ausgabedetails unterscheiden sich die Befehle?

2. Versuchen Sie, die Befehle `who` und `w -his` auszuführen.
    - Welche Informationen wurden aus der Ausgabe des Kommandos `w` mit den Optionen “no header” (`-h`) und “kurz” (`-s`) entfernt?
    - Welche Informationen wurden der Ausgabe des Kommandos `w` mit der Option “IP-Adresse” (`-i`) hinzugefügt?
3. In welcher Datei wird der Einweg-Passwort-Hash eines Benutzerkontos speichert?
4. Welche Datei enthält die Liste der Gruppen, in denen ein Benutzerkonto Mitglied ist? Wie könnte man eine Liste der Gruppen zusammenzustellen, in denen ein Benutzerkonto Mitglied ist?
5. Eine oder mehrere der folgenden Dateien sind standardmäßig für normale, nicht privilegierte Benutzer nicht lesbar. Welche?
    - `/etc/group`
    - `/etc/passwd`
    - `/etc/shadow`
    - `/etc/sudoers`
6. Wie würden Sie die Login-Shell des aktuellen Benutzers auf die Korn-Shell (`/usr/bin/ksh`) im nicht-interaktiven Modus ändern?
7. Warum befindet sich das Home-Verzeichnis des `root`-Benutzers nicht im Verzeichnis `/home`?

Zusammenfassung

In dieser Lektion haben wir die Linux-Benutzer- und Gruppendatenbanken sowie die wichtigsten Eigenschaften von Benutzern und Gruppen kennengelernt, einschließlich ihrer Namen und ihrer numerischen IDs. Wir haben zudem untersucht, wie das Passwort-Hashing unter Linux funktioniert und wie Benutzer den Gruppen zugeordnet werden.

All diese Informationen werden in den folgenden vier Dateien gespeichert, die die grundlegendsten lokalen Sicherheitszugangskontrollen auf einem Linux-System bieten:

`/etc/passwd`
: Alle systemlokalen POSIX-Attribute der Benutzerkonten, außer dem Passwort-Hash — für alle lesbar.

`/etc/group`
: Alle systemlokalen POSIX-Attribute der Gruppenkonten — für alle lesbar.

`/etc/shadow`
: Alle systemlokalen Benutzer-Passwort-Hashes (und Ablaufinformationen) — unlesbar für alle (nur ausgewählte Prozesse).

`/etc/sudoers`
: Alle systemlokalen Privilegien-Eskalationsinformationen/Zulassung durch den `sudo`-Befehl.

Die folgenden Befehle wurden in dieser Lektion behandelt:

`id`
: Echte (oder effektive) Benutzer- und Gruppen-IDs auflisten.

`last`
: Benutzer auflisten, die sich zuletzt angemeldet haben.

`who`
: Benutzer auflisten, die derzeit angemeldet sind.

`w`
: Ähnlich wie `who`, aber mit zusätzlichem Kontext.

`su`
: Wechselt zu einem anderen Benutzer mit einer Login-Shell oder führt Befehle als dieser Benutzer aus, indem man das Passwort dieses Benutzers übergibt.

`sudo`
: Switch User (oder Superuser) Do: Gibt, falls berechtigt, der aktuelle Benutzer sein eigenes Passwort ein (falls erforderlich), um die Berechtigungen zu erweitern.

`chsh`
: Ändert die Shell eines Benutzers.

`chfn`
: Ändert die Informationen des Benutzers im GECOS-Feld.