Sicherheitsgrundlagen in Linux

TECH STARTER

Agenda

1. Benutzerkonten und -information
2. Benutzer und Gruppen anlegen
3. Dateiberechtigungen/-eigentum
4. Besondere Verzeichnisse und Dateien

Benutzerkonten und -informationen

Sicherheit in Linux-Systemen

- ➔ Basiert auf UNIX-Zugriffskontrollen (3-stufiges Berechtigungsmodell, Verteilung auf Zugriffsrechte)
- ➔ Moderne Linux-Distributionen vereinfachen die Verwaltung von Benutzern
  - ◆ Kein manueller Eingriff des Systemadministrators

Benutzerkonten

- ➔ Sicherheitskonzept = Konzept der Zugangskontrolle
- ➔ Arten von Benutzerkonten (Accounts)
- ➔ Jeder Benutzer hat ein zugeordnetes Konto (Account)
  - ◆ Login-Informationen (Benutzername und Passwort)
  - ◆ Informationen wie und wo Benutzer mit dem System interagieren kann
- ➔ Berechtigungen + Zugriffskontrollen → Grenzen, in denen jeder Benutzer agiert

Identifier (UIDs/GIDs)

- ➔ User und Group Identifiers (UIDs/GIDs) = durchnummerierte Referenzen auf Accounts
- ➔ Früher 16-Bit-Ganzzahlen (0 - 65535)
- ➔ Heute 64-Bit-Ganzzahlen und Benutzer und Gruppen werden unabhängig voneinander gezählt
- ➔ Jeder Benutzer hat eine UID und eine primäre GID
  - ◆ Primäre GID kann für Benutzer einmalig sein oder viele Benutzer umfassen
- ➔ Neben der primären GID kann jeder Benutzer auch Mitglied anderer Gruppen sein
  - ◆ Standardmäßig: Jeder Benutzer einer Gruppe mit demselben Namen wie sein Benutzername und derselben GID wie seine UID zugeordnet

Identifier (UIDs/GIDs)

Wir erstellen uns einen neuen User newUser

```
sudo useradd newUser
```

Überprüfe mit `cat /etc/passwd | grep newUser`

(Informationen über Benutzernamen, UID, GID, Home-Verzeichnis, Standard-Shell)

Superuser-Account

- ➔ Superuser root UID 0
- ➔ Superuser = “Systemadministrator”
- ➔ Unbegrenzten Zugriff/Kontrolle über das System und Benutzer
- ➔ Standardgruppe ist root mit GID 0
- ➔ Home-Verzeichnis → dediziertes Top-Level-Verzeichnis /root

Standard-Benutzerkonten

- ➔ Alle Accounts außer root = reguläre Benutzerkonten
- ➔ Typische Eigenschaften
  - ◆ UIDs ab 1000 (4-Stellen, bei alten Systemen manchmal 500)
  - ◆ Definiertes Home-Verzeichnis (Unterverzeichnis von /home)
  - ◆ Definierte Login-Shell, also Standardshell. Meist Bash.
  - ◆ Achtung: Wenn keine gültige Shell zugeordnet ist, kann Benutzer keine interaktive Shell öffnen. Meist wird das mit /sbin/nologin als ungültige Shell gesetzt. Kann sinnvoll sein, wenn Benutzer nur für andere Dienste als Konsolen- oder SSH-Zugriff authentifiziert wird, z.B. Secure FTP (sftp)
- ➔ Benutzerkonto/User Account = reguläre Benutzerkonten
- ➔ Systemkonto/System Account = Linux-Benutzerkonto vom Typ Systembenutzerkonto

System Accounts

- ➔ Werden bei Systeminstallation erstellt für Programme/Dienste, die nicht als Superuser laufen, z.B. alles Betriebssystemspezifisches
- ➔ Typische Eigenschaften:
  - ◆ UIDs unter 100 (2-stellig o. Zwischen 500 und 1000)
  - ◆ kein dediziertes Home-Verzeichnis bzw. ein Verzeichnis, das normalerweise nicht unter /home liegt
  - ◆ keine gültige Login-Shell (/sbin/nologin) mit seltenen Ausnahmen
- ➔ Meistens benötigen sie keiner Anmeldung. Keine gültige Login-Shell, da dies ein Sicherheitsrisiko wäre.
- ➔ Viele Prozesse, die System Accounts gehören und von diesen ausgeführt werden, zweigt Systemmanagement in eine eigene Umgebung ab, die mit dem angegebenen System Account läuft.
- ➔ eingeschränkte oder keine Berechtigungen

Service Accounts

- ➔ werden bei Installation/Konfiguration von Diensten erstellt (ähnlich wie Systemkonten für Programme/Dienste), die nicht als Superuser ausgeführt werden
- ➔ häufig System Accounts = Service Accounts
- ➔ Eigenschaften
  - ◆ Home-Verzeichnis außerhalb von /home (Service Accounts haben eher eins als System Accounts)
  - ◆ keine gültige Login-Shell
  - ◆ UID/GID > 1000 (4-und-mehr-stellig), aber kein Standard o. regulärer Benutzer-Account (Unterschied zu Systemaccounts, wo die UID/GID < 100 bzw. < 500-1000 ist). Manchmal haben reservierte Service Accounts eine UID < 100, die man auch als Systemkonto betrachten könnte, obwohl sie nicht bei Systeminstallation erstellt werden. Beispiel: Apache-Webserver bei Fedora-basierten Linux-Distributionen (Red Hat) die UID/GID 48 und damit ein Systemkonto, obwohl er ein Home-Verzeichnis hat (/usr/share/httpd o. /var/www/html)
- ➔ Merke:
  - ◆ Systemkonten mit UID < 1000
  - ◆ Benutzerkonten UID > 1000
  - ◆ Servicekonto UID > 1000

Login-Shells

- ➔ Einige Konten haben eine Login-Shell (andere aus Sicherheitsgründen keine)
- ➔ Standard-Login-Shell meist Bash
  - ◆ Manchmal können auch C-Shell (csh), Korn-Shell (ksh) oder Z-Shell (zsh) verfügbar sein
- ➔ Benutzer kann seine Login-Shell mit dem Befehl chsh ändern
  - ◆ Befehl im interaktiven Modus mit Eingabeaufforderung, wo man den vollständigen Pfad zum Binary der gewünschten Shell angibt

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

```
$ chsh

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

Login-Shells

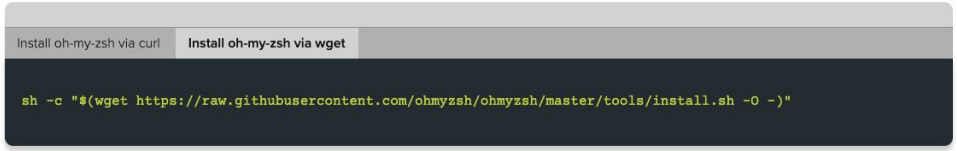
- ➔ z-Shell installieren mit sudo apt install zsh
- ➔ gib which zsh ein
- ➔ chsh und dann /usr/bin/zsh eingeben
- ➔ jetzt überprüfen mit cat /etc/passwd | grep <username>
- ➔ Alternativ ein neues Fenster vom Tab aufmachen und es erscheint eine Aufforderungen, etwas einzugeben
  - ◆ Wir haben noch keine .zshrc angelegt..
- ➔ Wir skippen das Ganze und können unsere Standard-Shell auf Bash zurückstellen
- ➔ Alles wieder fein

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

```
$ chsh

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

Login-Shells - Ausflug



- ➔ Wenn wir das Aussehen von unserer Shell verändern wollen, können wir z.B. das ohmyzsh-Plugin für zsh installieren (https://ohmyz.sh/)
- ➔ Installiere mit dem wget-Befehl. Ihr erhaltet dann eine erfolgreiche Meldung.
- ➔ Hiermit können wir unsere Shell im Aussehen deutlich verändern. Es gibt verschiedene Themes. Die Themes könnt ihr euch in der offiziellen Dokumentation ansehen.
- ➔ Wir setzen Themes in der ~/.zshrc, die wir z.B. mit vim ~/.zshrc öffnen können..

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

```
$ chsh

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

Home-Verzeichnisse

- ➔ Die meisten Accounts haben ein definiertes Home-Verzeichnis
- ➔ einziger Ort, wo dieser Benutzer sicher Schreibrechte hat
  - ◆ Ausnahme z.B. temporäre Dateisystembereiche
- ➔ Einige Accounts sind bewusst so eingerichtet, dass sie aus Sicherheitsgründen keinen Schreibzugriff auf ihr eigenes Home-Verzeichnis haben

Benutzerinformationen abfragen

- ➔ wird häufig gemacht in der Praxis: In einigen Fällen müssen Benutzer den Benutzer wechseln und die Berechtigung erweitern, um priviligierte Aufgaben auszuführen.
- ➔ Mit id können Informationen zu einem Benutzer angezeigt werden
  - 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.

```
$ 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
```

Benutzerinformationen abfragen

- ➔ Mit last lassen wir uns die letzten Anmeldungen von Benutzern am System anzeigen lassen
  - ◆ 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.
- ➔ lastb listet zuletzt fehlgeschlagene Anmeldeversuche auf

```
$ 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)
    ...
```

Benutzerinformationen abfragen

- ➔ who und w listen nur die aktiven Anmeldungen am System auf
- ➔ 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.
- ➔ w listet auch folgendes auf:
  - ◆ 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
  - ◆ Auswahl der gesamten CPU-Nutzungszeiten (IDLE, JCPU und PCPU)
  - ◆ Aktueller Prozess (-bash) — die gesamte CPU-Nutzung dieses Prozesses ist der letzte Eintrag (PCPU).

```
$ 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
```

Berechtigungen erweitern

- ➔ In einer idealen Welt: Benutzer müssen Dateirechte nicht erweitern. Das System würde immer funktionieren.
- ➔ Least-Privilege-Sicherheitsmodell
- ➔ su und sudo erlauben eine Rechteerweiterung bei Bedarf

Benutzerwechsel su

- ➔ su - = Vergabe von root-Rechten an den aktuellen Benutzer
- ➔ su - <username> = Benutzerwechsel auf beliebigen Nutzer (nicht empfohlen)
- ➔ root-Passwort muss eingegeben werden, dann hat der Benutzer eine Superuser-Shell (siehe das #)
- ➔ Schlechte Sicherheitspraxis (Passwort-Weitergabe) → wenig Verwendung des su-Befehls in der Praxis
- ➔ Achtung! Nicht so leichtfertig verwenden
  - ◆ Sitzung eines regulären Benutzers kompromittiert wurde, dann könnte das Passwort des Superusers abgefangen werden. Daher wollen wir eher “Switch User Do” bzw. “Superuser Do” machen..

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

Benutzerwechsel sudo

- ➔ z.B. wollen wir die Seriennummer unserer Systemplatine auslesen. Der Zugriff wird verweigert, da diese Information als privilegiert gekennzeichnet ist.
- ➔ Indem man sudo nutzt und das eigene Passwort eingibt, authentifiziert sich der Benutzer, wer er ist.
- ➔ Wenn er in der sudoers-Konfiguration autorisiert wurde, diesen Befehl auszuführen, wird es funktionieren.
- ➔ Man ist standardmäßige für eine bestimmte Zeitspanne authentifiziert (wird konfiguriert durch Systemadministrator)

```
$ 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/
```

Access-Control-Dateien

- ➔ Fast alle Betriebssysteme speichern Zugriffskontrollen an bestimmten Orten
- ➔ Typischerweise Textdateien im Verzeichnis /etc, in dem Systemkonfigurationsdateien liegen.
- ➔ Standardmäßig für jeden lesbar, aber nur durch root beschreibbar

- ➔ /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. Jeder Benutzer kann lesen und root darf schreiben.
- ➔ /etc/group Diese Datei enthält grundlegende Informationen über alle Benutzergruppen auf dem System, wie Gruppenname, GID und Mitglieder. Jeder Benutzer kann lesen und root darf schreiben.
- ➔ /etc/shadow Hier werden die Benutzerpasswörter gespeichert, die aus Sicherheitsgründen gehasht sind. Hier darf root lesen.
- ➔ /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. Hier darf root lesen.

Access-Control-Dateien

- ➔ Es gibt Dateien, die mit der grundlegenden Rechteerweiterung auf Linux-Systemen zu tun haben (z.B. su und sudo). Diese sind nur für 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. Ist eine Textdatei, sollte aber nie direkt bearbeitet werden, sondern nur mit dem Werkzeug visudo

/etc/passwd

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

- ➔ 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.
- ➔ Syntax:
  - ◆ 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.

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

/etc/passwd

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

- ➔ 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.
- ➔ Syntax:
  - ◆ 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.

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

/etc/passwd

```
USERNAME:PASSWORD:UID:GID:GECOS:HOMEDIR:SHELL
emma:x:1000:1000:Emma Smith,42 Douglas St,555.555.5555:/home/emma:/bin/bash
```

➔ GECOS-Feld

- 3 oder mehr Felder (durch Komma getrennt → CSV (Comma Separated Values))
- NAME, LOCATION, CONTACT
  - 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.
- und noch weitere Informationen…
- Informationen ändern mit chfn

```
$ 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
```

```
NAME:PASSWORD:GID:MEMBERS

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

/etc/group

- ➔ Felder mit : abgegrenzt
- ➔ Grundlegende Informationen über Gruppen auf dem System
- ➔ “Group File” “Gruppendatei”
  - ◆ 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.
- ➔ Wenn Gruppe primäre Gruppe des Benutzers ist, muss der Benutzer nicht aufgeführt werden (sonst redundant)
- ➔ Passwort für Gruppe liegt gehashed in der Datei /etc/gshadow

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

```
emma:$6$nP532JDDogQYZF8I$bjFNh9eT1xpb9/n6pmjlIwgu7hGjH/eytSdttbmVv0MlyTMFgBIXESFNUmTo9EGxxH1
OT1HGQzR0so4n1npbE0:18064:0:99999:7:::
```

/etc/shadow

- ➔ Felder durch : abgegrenzt
- ➔ “Shadow Datei”
  - 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”.
- ➔ Kann nur vom Superuser ausgewählten Kernsystem-Authentifizierungsdiensten gelesen werden

/etc/shadow

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

```
emma:$6$nP532JDDogQYZF8I$bjFNh9eT1xpb9/n6pmjlIwgu7hGjH/eytSdttbmVv0MlyTMFgBIXESFNUmTo9EGxxH1
OT emma:$6$nP532JDDogQYZF8I$bjFNh9eT1xpb9/n6pmjlIwgu7hGjH/eytSdttbmVv0MlyTMFgBIXESFNUmTo9EGxxH1
  OT1HGQzR0so4n1npbE0:18064:0:99999:7:::
```

- ➔ Felder durch : abgegrenzt
- ➔ “Shadow Datei”
  - ◆ 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”.
- ➔ Kann nur vom Superuser ausgewählten Kernsystem-Authentifizierungsdiensten gelesen werden

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

Hash-Funktion

- ➔ Es gibt verschiedene Authentifizierungslösungen
- ➔ Elementare Methode der Passwortspeicherung → einseitige Hash-Funktion
- ➔ So wird das Passwort niemals in Klartext auf einem System gespeichert, da die Hash-Funktion nicht umkehrbar ist
  - ◆ Wir können ein Passwort in einen Hash umwandeln, aber idealerweise ist es nicht möglich, einen Hash wieder in ein Passwort umzuwandeln
- ➔ Man benötigt erst eine Brute-Force-Methode, um alle Kombinationen eines Passworts zu hashen, bis eines übereinstimmt. Um diese Wahrscheinlichkeit zu verringern, verwenden Linux-Systeme ein zufälliges Salz (“Salt”) auf jedem Passwort-Hash für einen Benutzer
  - ◆ Hash auf einem Linux-System ist i.d.R. nicht dasselbe wie auf einem anderen Linux-System, auch wenn Passwort gleich

Verschiedene Formen des Passworts

- In der Datei /etc/shadow kann das Passwort verschiedene Formen haben
  - !! 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.
- Hash-Funktion, Hash-Salt und Hash-String werden durch ein Dollar-Symbol $ eingeleitet und abgegrenzt. Hash-Salt ist 8-16 Zeichen lang. Meistens werden die folgenden verwendet:
  - $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.

/etc/gshadow

➔ Datei mit vier durch Doppelpunkte getrennten Feldern, die verschlüsselte Gruppenpasswörter enthalten.

Aufgaben

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

1. In welchen Dateien werden die folgenden Attribute gespeichert?
   a. UID und GID
   b. Gruppen
2. In welcher Datei wird zusätzlich das Benutzerpasswort gespeichert?

Aufgaben

1. Welche der folgenden Arten von Kryptographie wird standardmäßig verwendet, um Passwörter lokal auf einem Linux-System zu speichern?

Aufgaben

1. Wenn ein Konto eine UID unter 1000 hat, um welche Art von Konto handelt es sich?
2. Wie erhalten wir eine Liste der aktiven Logins auf unserem System und deren Anzahl?

Aufgaben

1. Was sehen wir hier alles in der Ausgabe?

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

Aufgaben

1. Vergleiche die Ergebnisse von last mit w und who
2. In welcher Datei wird der Einweg-Passwort-Hash eines Benutzerkontos gespeichert?
3. Welche Datei enthält die Liste der Gruppen, in denen ein Benutzerkonto Mitglied ist? Wie könnte man eine Liste zusammenstellen, in denen ein Benutzerkonto Mitglied ist?
4. Wie können wir Login-Shell des aktuellen Benutzers im nicht-interaktiven Modus ändern?

Hinzufügen von Benutzerkonten

- useradd = neues Benutzerkonto hinzufügen
- userdel = Benutzerkonto löschen
- Nach dem Anlegen mit passwd <username> ein Passwort setzen
- Brauchen dafür root-Rechte
- Optionen:
  - -c Erstellt ein neues Benutzerkonto mit Kommentaren (z.B. dem vollen Namen).
  - -d Erstellt ein neues Benutzerkonto mit dem angegebenen Home-Verzeichnis.
  - -e Erstellt ein neues Benutzerkonto, das zu dem angegebenen Datum deaktiviert wird.
  - -f Erstellt ein neues Benutzerkonto; innerhalb der angegebenen Anzahl von Tagen muss der Benutzer nach Ablauf des Passworts das Passwort aktualisieren.
  - -g Erstellt ein neues Benutzerkonto mit der angegebenen GID.
  - -G Erstellt ein neues Benutzerkonto, das den angegebenen sekundären Gruppen hinzufügt wird.
  - -m Erstellt ein neues Benutzerkonto mit dem angegebenen Home-Verzeichnis. -M Erstellt ein neues Benutzerkonto ohne Home-Verzeichnis. -s Erstellt ein neues Benutzerkonto mit der angegebenen Login-Shell. -u Erstellt ein neues Benutzerkonto mit der angegebenen UID.

```
# useradd frank
```

```
# passwd frank
Changing password for user frank.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

Hinzufügen von Benutzerkonten

- ➔ Nach dem Erstellen mit id <username> und groups <username> alles überprüfen
- ➔ Achtung: Datei /etc/login.defs überprüfen und evtl. editieren für die Konfigurationsparameter
  - ◆ z.B. Bereich der UIDs und GIDs festlegen, die neuen Benutzer- und Gruppen-Accounts zugewiesen werden
  - ◆ Festlegen, dass -m nicht benutzt werden muss, um das Home-Verzeichnis des neuen Benutzers zu erstellen
  - ◆ Automatisch eine Gruppe für jeden Benutzer anlegen

```
# useradd frank
```

```
# passwd frank
Changing password for user frank.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

Löschen von Benutzerkonten

```
# userdel -r frank
```

- ➔ userdel <username> löscht den Benutzeraccount
- ➔ aktualisiert auch die Account-Datenbanken und löscht alle Einträge, die sich auf den angegebenen Benutzer beziehen.
- ➔ -r entfernt auch das Home-Verzeichnis des Benutzers und seinen gesamten Inhalt sowie den Mail-Spool des Benutzers
- ➔ Wir brauchen auch hier root-Rechte

Skeleton-Verzeichnis

- ➔ Wenn wir einen neuen Benutzer anlegen und das Home-Verzeichnis dafür erstellen, wird das neu erstellte Home-Verzeichnis mit Dateien und Ordnern gefüllt, die aus dem Skeleton-, also Skelett-Verzeichnis (/etc/skel) kopiert werden
- ➔ Idee:
  - ◆ Systemadministrator möchte neue Benutzer hinzufügen, die alle dieselben Dateien und Verzeichnisse in ihrem Home-Verzeichnis haben. Wenn Dateien und Ordner angepasst werden sollen, die im Home-Verzeichnis landen sollen, dann müssen wir diese neuen zum Skeleton-Verzeichnis hinzufügen
- ➔ Die Profildateien im Skeleton-Verzeichnis sind versteckte Dateien. Wir brauchen also ein ls -a, um alle aufzulisten

Hinzufügen und Löschen von Gruppen

- ➔ groupadd <gruppenname> und groupdel <gruppenname>
- ➔ -g gibt direkt bestimmte GID an

```
# groupadd -g 1090 developer
```

```
# groupdel developer
```

passwd

- ➔ Passwort eines Benutzers ändern
- ➔ Jeder kann sein eigenes Passwort ändern, root kann alle ändern
- ➔ Optionen (nur für root):
  - ◆ -d Löscht das Passwort eines Benutzerkontos (setzt also ein leeres Passwort und macht es zu einem passwortlosen Account).
  - ◆ -e Zwingt den Benutzer, das Passwort zu ändern.
  - ◆ -l Sperrt das Benutzerkonto (dem verschlüsselten Passwort wird ein Ausrufezeichen vorangestellt).
  - ◆ -u Entsperrt das Benutzerkonto (entfernt das Ausrufezeichen). -S Gibt Informationen über den Passwortstatus für ein bestimmtes Konto aus

Aufgabe

1. Was ist die UID und GID von carol?
2. Welche Shell ist für dave und henry eingestellt?
3. Wie ist der Name der Hauptgruppe von henry?
4. Welche sind die Mitglieder der Gruppe web_developer? Welche von ihnen sind Gruppenadministratoren?
5. Welcher Benutzer kann sich nicht in das System einloggen?
6. Welcher Benutzer sollte das Passwort bei der nächsten Anmeldung im System ändern?
7. Wie viele Tage müssen vergehen bis eine Passwortänderung von carol erforderlich ist?

```
# cat /etc/passwd | tail -3
dave:x:1050:1050:User Dave:/home/dave:/bin/bash
carol:x:1051:1015:User Carol:/home/carol:/bin/sh
henry:x:1052:1005:User Henry:/home/henry:/bin/tcsh
# cat /etc/group | tail -3
web_admin:x:1005:frank,emma
web_developer:x:1010:grace,kevin,christian
dave:x:1050:
# cat /etc/shadow | tail -3
dave:$6$AbCdEfGh123456789A1b2C3D4e5F6G7h8i9:0:20:90:7:30::
carol:$6$q1w2e3r4t5y6u7i8AbcDeFgHiLmNoPqRsTu:18015:0:60:7:::
henry:!$6$123456789aBcDeFgHa1B2c3d4E5f6g7H8I9:18015:0:20:5:::
# cat /etc/gshadow | tail -3
web_admin:!:frank:frank,emma
web_developer:!:kevin:grace,kevin,christian
dave:!::
```

Übung

➔ https://docs.google.com/document/d/1PT0qKTrGfLE98WTXhPgB66HUVMu1rX1sD5y1FS_L6s0/edit?usp=sharing

Sicherheitsgrundlagen in Linux

Dateiberechtigungen

Dateiberechtigungen

- ➔ Linux als Mehrbenutzersystem muss nachvollziehen, wem welche Dateien gehören und ob ein Benutzer Aktionen mit bestimmten Dateien ausführen darf oder nicht
- ➔ 3-stufiges Berechtigungssystem. Jede Datei auf der Festplatte gehört einem Benutzer und einer Benutzergruppe und hat 3 Berechtigungsgruppen
  - ◆ Eigentümer
  - ◆ Gruppe
  - ◆ Alle anderen

Informationen über Dateien und Verzeichnisse abfragen

```
$ ls
Another_Directory  picture.jpg  text.txt
```

- ➔ Mit Befehl ls Inhalte eines beliebigen Verzeichnisses auflisten. In der Grundform erhalten wir nur Datei- und Verzeichnisname.
- ➔ Mit ls -l → ausführlichere Informationen (Typ, Größe, Eigentümer, uvm.)

```
$ ls -l
total 536
drwxrwxr-x 2 carol carol   4096 Dec 10 15:57 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

Informationen über Dateien und Verzeichnisse abfragen

```
$ ls
Another_Directory  picture.jpg  text.txt
```

- ➔ Mit Befehl ls Inhalte eines beliebigen Verzeichnisses auflisten. In der Grundform erhalten wir nur Datei- und Verzeichnisname.
- ➔ Mit ls -l → ausführlichere Informationen (Typ, Größe, Eigentümer, uvm.)

```
$ ls -l
total 536
drwxrwxr-x 2 carol carol   4096 Dec 10 15:57 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

- ➔ Die 1. Spalte zeigt den Dateityp und Zugriffsrechte, z.B. drwxrwxr-x
  - ◆ Das erste Zeichen, d, zeigt den Dateityp an.
  - ◆ Die nächsten drei Zeichen, rwx, zeigen die Rechte des Besitzers der Datei an, auch user oder u genannt.
  - ◆ Die nächsten drei Zeichen, rwx, zeigen die Rechte der Gruppe an, der die Datei gehört, auch als g bezeichnet.
  - ◆ Die letzten drei Zeichen, r-x, zeigen die Rechte für jeden anderen an, auch als other oder o bezeichnet.

Informationen über Dateien und Verzeichnisse abfragen

```
$ ls
Another_Directory  picture.jpg  text.txt
```

- ➔ Mit Befehl ls Inhalte eines beliebigen Verzeichnisses auflisten. In der Grundform erhalten wir nur Datei- und Verzeichnisname.
- ➔ Mit ls -l → ausführlichere Informationen (Typ, Größe, Eigentümer, uvm.)

```
$ ls -l
total 536
drwxrwxr-x 2 carol carol   4096 Dec 10 15:57 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

- ➔ Die 2. Spalte gibt Anzahl der Hardlinks an, die auf diese Datei zeigen.
  - ◆ Für ein Verzeichnis bedeutet das “Anzahl der Unterverzeichnisse” + Link auf sich selbst (.) und das übergeordnete Verzeichnis (..)

Informationen über Dateien und Verzeichnisse abfragen

```
$ ls
Another_Directory  picture.jpg  text.txt
```

- ➔ Mit Befehl ls Inhalte eines beliebigen Verzeichnisses auflisten. In der Grundform erhalten wir nur Datei- und Verzeichnisname.
- ➔ Mit ls -l → ausführlichere Informationen (Typ, Größe, Eigentümer, uvm.)

```
$ ls -l
total 536
drwxrwxr-x 2 carol carol   4096 Dec 10 15:57 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

- ➔ Die 3. und 4. Spalte zeigen Eigentümer bzw. Benutzer und Gruppe an, die diese Datei besitzen
- ➔ Die 5. Spalte zeigt die Dateigröße in Bytes an
- ➔ Die 6. Spalte den Zeitstempel, wann die Datei zuletzt geändert wurde
- ➔ Die 7. Spalte zeigt den Dateinamen

Informationen über Dateien und Verzeichnisse abfragen

```
$ ls
Another_Directory  picture.jpg  text.txt
```

- ➔ Mit Befehl ls Inhalte eines beliebigen Verzeichnisses auflisten. In der Grundform erhalten wir nur Datei- und Verzeichnisname.
- ➔ Mit ls -l → ausführlichere Informationen (Typ, Größe, Eigentümer, uvm.)

```
$ ls -lh
total 1,2G
drwxrwxr-x 2 carol carol 4,0K Dec 10 17:59 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
-rw------- 1 carol carol 528K Dec 10 10:43 picture.jpg
---xr-xr-x 1 carol carol   33 Dec 11 10:36 test.sh
-rwxr--r-- 1 carol carol 1,9K Dec 20 18:13 text.txt
-rw-rw-r-- 1 carol carol 2,6M Dec 11 22:14 Zipped.zip
```

- ➔ Mit -h können wir die Ausgabe im menschenlesbaren Format anzeigen
  - ◆ Dateien < Kilobyte = in Bytes angezeigt
  - ◆ Dateien > Kilobyte && < Megabyte = Mit K in Kilobyte
  - ◆ usw. für Megabyte (M) und Gigabyte (G)

Versteckte Dateien sehen

- ➔ Alle die mit . beginnen, sind in Linux versteckt
- ➔ Sehen mit ls -a
- ➔ . und .. Sonderfälle

```
$ ls -l -a
total 544
drwxrwxr-x 3 carol carol   4096 Dec 10 16:01 .
drwxrwxr-x 4 carol carol   4096 Dec 10 15:56 ..
drwxrwxr-x 2 carol carol   4096 Dec 10 17:59 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
-rw-r--r-- 1 carol carol      0 Dec 10 16:01 .thisIsHidden
```

Dateitypen verstehen

- ➔ - (normale Datei) Eine Datei kann Daten jeglicher Art enthalten. Dateien können verändert, verschoben, kopiert und gelöscht werden
- ➔ d (Verzeichnis) Ein Verzeichnis enthält andere Dateien oder Verzeichnisse und hilft, das Dateisystem zu organisieren. Technisch gesehen sind Verzeichnisse eine besondere Art von Dateien.
- ➔ l (Soft Link) Diese “Datei” ist ein Zeiger auf eine andere Datei oder ein anderes Verzeichnis im Dateisystem. Neben diesen gibt es drei weitere Dateitypen, die Sie zumindest kennen sollten, die aber nicht Thema dieser Lektion sind:
- ➔ b (Block Device) Diese Datei steht für ein virtuelles oder physisches Gerät, normalerweise Festplatten oder andere Arten von Speichergeräten. Zum Beispiel könnte die erste Festplatte im System durch /dev/sda dargestellt werden.
- ➔ c (Character Device) Diese Datei steht für ein virtuelles oder physisches Gerät. Terminals (wie das Hauptterminal auf /dev/ttyS0) und serielle Schnittstellen sind gängige Beispiele für Character Devices.
- ➔ s (Socket) Sockets dienen als “Leitungen”, die Informationen zwischen zwei Programmen weiterleiten.

Berechtigungen auf Dateien

- ➔ Mit ls -l konnten wir jeweils Dateiberechtigungen direkt nach dem Dateityp als 3 Gruppen von je drei Zeichen (r,w und x) sehen
  - r Steht für “read”, also “lesen”, und hat einen Oktalwert von 4 (wir werden in Kürze auf Oktale eingehen). Das bedeutet die Berechtigung, eine Datei zu öffnen und ihren Inhalt zu lesen.
  - w Steht für “write”, also “schreiben”, und hat einen Oktalwert von 2. Das bedeutet die Berechtigung, eine Datei zu bearbeiten oder zu löschen.
  - x Steht für “execute”, also “ausführen”, und hat einen oktalen Wert von 1. Das bedeutet, dass die Datei als ausführbare Datei oder als Skript ausgeführt werden kann.
- ➔ z.B. Datei mit rw- kann zwar gelesen und geschrieben, aber nicht ausgeführt werden

```
$ ls -l
total 536
drwxrwxr-x 2 carol carol   4096 Dec 10 15:57 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

Berechtigungen auf Verzeichnissen

➔ Mit ls -l konnten wir jeweils Dateiberechtigungen direkt nach dem Dateityp als 3 Gruppen von je drei Zeichen (r,w und x) sehen

- r Steht für “read” und hat einen oktalen Wert von 4. Das bedeutet die Berechtigung, den Inhalt des Verzeichnisses zu lesen, wie z.B. Dateinamen. Aber es bedeutet nicht die Berechtigung, die Dateien selbst zu lesen.
- w Steht für “write” und hat den oktalen Wert 2. Das bedeutet die Berechtigung, Dateien in einem Verzeichnis zu erstellen oder zu löschen oder deren Namen, Rechte und Besitzer zu ändern. Wenn ein Benutzer die Schreibberechtigung für ein Verzeichnis hat, kann er die Berechtigungen jeder Datei in dem Verzeichnis ändern, auch wenn er keine Rechte auf die Datei hat oder die Datei einem anderen Benutzer gehört.
- x Steht für “execute” und hat einen oktalen Wert von 1. Das bedeutet die Berechtigung, in ein Verzeichnis zu wechseln, aber nicht, seine Dateien aufzulisten (dafür ist die r-Berechtigung erforderlich).

```
$ ls -ld Another_Directory/
d--xr-xr-x 2 carol carol 4,0K Dec 20 18:46 Another_Directory
```

Berechtigungen auf Verzeichnissen

- ➔ Beispiel: Wir haben ein Verzeichnis mit folgenden Berechtigungen
- ➔ In dem Verzeichnis liegt ein Shell-Skript hello.sh mit den Rechten
- ➔ Angenommen wir sind der Benutzer carol. Wenn wir versuchen den Inhalt von Another_Directory versuchen aufzulisten, erhalten wir eine Fehlermeldung, da der Benutzer keine Leseberechtigung für dieses Verzeichnis hat
- ➔ carol hat jedoch Ausführungsrechte, d.h. Man darf das Verzeichnis betreten und auf Dateien innerhalb des Verzeichnisses zugreifen, sofern man die richtigen Rechte für die jeweilige Datei hat. Das Skript können wir ausführen.

```
$ ls -ld Another_Directory/
d--xr-xr-x 2 carol carol 4,0K Dec 20 18:46 Another_Directory
```

```
-rwxr-xr-x 1 carol carol 33 Dec 20 18:46 hello.sh
```

```
$ ls -l Another_Directory/
ls: cannot open directory 'Another_Directory/': Permission denied
```

```
$ sh Another_Directory/hello.sh
Hello LPI World!
```

3-stufige Berechtigungssystem

- ➔ Rechte werden in der folgenden Reihenfolge vergeben:
  - Zuerst für den Eigentümer der Datei
  - Dann für die besitzende Gruppe
  - Dann für andere Benutzer
- ➔ Überprüfung erfolgt genauso:
  - Zuerst wird geprüft, ob der aktuelle Benutzer die Datei besitzt. Trifft das zu, wird der erste Satz an Rechten angewandt.
  - Andernfalls prüfe ob aktueller Benutzer zur Gruppe gehört. Dann wird der zweite Satz angewandt.
  - In jedem anderen Fall wird der dritte Satz von Rechten angewandt

Ändern von Dateiberechtigungen

- ➔ chmod
- ➔ Rechte für eine Datei ändern
- ➔ Brauchen 2 Parameter
  - ◆ Erster beschreibt welche Rechte geändert werden soll
  - ◆ Zweiter zeigt auf die Datei oder das Verzeichnis, an dem die Änderung vorgenommen wird.
- ➔ Rechte zum Ändern können auf zwei verschiedene Arten oder Modes beschrieben werden.
  - ◆ Symbolic Mode: besser um einzelne Berechtigungen hinzuzufügen oder zu widerrufen, ohne andere im Set zu ändern.
  - ◆ Numeric Mode: leichter zu merken und schneller zu benutzen wenn wir alle Berechtigungswerte auf einmal setzen

```
$ chmod ug+rw-x,o-rwx text.txt
```

```
$ chmod 660 text.txt
```

```
-rw-rw---- 1 carol carol  765 Dec 20 21:25 text.txt
```

Symbolic Mode

- ➔ Gib zuerst an, welche Berechtigungen geändert werden sollen
  - u Benutzer
  - g Gruppe
  - o andere
  - a für alle zusammen
- ➔ Dann sagen, was getan werden soll
  - + Berechtigungen erteilen
  - - Berechtigung widerrufen
  - = Berechtigung auf bestimmten Wert setzen
- ➔ Zuletzt sagen, welche Rechte man beeinflussen möchte
  - r Lesen
  - w Schreiben
  - x Ausführen

```
$ ls -l text.txt
-rw-r--r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

```
$ chmod g+w text.txt
```

```
$ ls -l text.txt
-rw-rw-r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

```
$ chmod u-r text.txt
$ ls -l text.txt
--w-rw-r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

```
$ chmod u+rwx,g-x text.txt
$ ls -lh text.txt
-rwxrw-rw- 1 carol carol 765 Dec 20 21:25 text.txt
```

```
$ chmod -R u+rwx Another_Directory/
```

-R für rekursiven Mode: “Rekursiv Benutzer gewähren, Lese-, Schreib- und Ausführungsrechte
Aber ACHTUNG!

Numeric Mode

- ➔ Dreistelliger numerischer Wert in oktaler Notation (Basis 8)
- ➔ Jede Berechtigung hat entsprechenden Wert
  - ◆ Lesen r hat 4
  - ◆ Schreiben w hat 2
  - ◆ Ausführen x hat 1
  - ◆ Keine Berechtigung ist 0
  - ◆ z.B. rwx = 7 (4+2+1) und r-x = 5 (4+0+1)
- ➔ Selbes Prinzip
  - ◆ 1. Zahl steht für den Benutzer u
  - ◆ 2. Zahl steht für die Gruppe g
  - ◆ 3. Zahl steht für andere o

```
$ chmod 660 text.txt
$ ls -l text.txt
-rw-rw---- 1 carol carol 765 Dec 20 21:25 text.txt
```

Wann was anwenden?

- ➔ Ich möchte Berechtigungen auf einen bestimmten Wert ändern → Numeric Mode
- ➔ Ich möchte einen bestimmten Wert ändern, unabhängig von den aktuellen Berechtigungen für die Datei → Symbolic Mode

```
$ chmod 660 text.txt
$ ls -l text.txt
-rw-rw---- 1 carol carol 765 Dec 20 21:25 text.txt
```

Ändern des Dateibesitzes

- ➔ chown username:groupname filename = Besitzverhältnisse einer Datei oder eines Verzeichnisses verändern
- ➔ Der Benutzer, der die Datei besitzt muss nicht der Gruppe angehören, die die Datei besitzt.
- ➔ Nur Benutzer ändern: chown username filename
- ➔ Nur Gruppe ändern: chown :groupname filename
- ➔ Alternative: chgrp groupname filename verändert nur die Gruppe

```
$ ls -l text.txt
-rw-rw---- 1 carol carol 1881 Dec 10 15:57 text.txt
```

```
$ chown carol:students text.txt
$ ls -l text.txt
-rw-rw---- 1 carol students 1881 Dec 10 15:57 text.txt
```

Gruppen abfragen

- ➔ groups fragt ab, welche Gruppen auf dem System existieren
- ➔ groups username zeigt an, zu welchen Gruppen ein Benutzer gehört
- ➔ sudo groupmems -g groupname -l listet alle Mitglieder einer Gruppe auf

```
$ groups
carol students cdrom sudo dip plugdev lpadmin sambashare
```

```
$ groups carol
carol : carol students cdrom sudo dip plugdev lpadmin sambashare
```

```
$ sudo groupmems -g cdrom -l
carol
```

Spezielle Berechtigungen

- ➔ Neben den Lese-, Schreib- und Ausführungsrechten für Benutzer, Gruppen und anderen kann jede Datei drei weitere Berechtigungen haben.
- ➔ Angabe im Symbolic und Numeric Mode möglich

Spezielle Berechtigungen - Sticky Bit

- ➔ auch Restricted Deletion Flag genannt
- ➔ oktaler Wert 1 und im Symbolic Mode mit t in den Rechten
- ➔ Das gilt nur für Verzeichnisse, um Benutzer daran zu hindern, eine Datei in einem Verzeichnis zu entfernen oder umzubenennen, wenn man die Datei oder das Verzeichnis nicht besitzt.
- ➔ Verzeichnisse mit einem Sticky Bit zeigen ein t, das das x in andere ersetzt
- ➔ Im Numeric Mode gibt es eine vierstellige Notation, wobei die erste Ziffer die spezifische Berechtigung darstellt.
- ➔ Bei farblichen Terminal: Verzeichnis wird in schwarzer Schrift mit blauem Hintergrund angezeigt

```
$ ls -ld Sample_Directory/
drwxr-xr-t 2 carol carol 4096 Dec 20 18:46 Sample_Directory/
```

```
$ chmod 1755 Another_Directory
$ ls -ld Another_Directory
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
```

Spezielle Berechtigungen - Set GID

- ➔ Set GID o. SGIT o. Set Group ID Bit
- ➔ oktaler Wert 2 und im Symbolic Mode s und repräsentiert Gruppen-Berechtigungen
- ➔ Kann auf ausführbare Dateien oder Verzeichnisse angewandt werden
- ➔ Bei ausführbaren Dateien gewährt es dem Prozess der aus der Ausführung der Datei resultiert, Zugriff auf die Berechtigungen der Gruppe, der die Datei gehört. Auf Verzeichnissen angewendet, erbt jede Datei oder jedes Unterverzeichnis, die Gruppe vom übergeordneten Verzeichnis
- ➔ Bei farblichen Terminal: Verzeichnis wird in schwarzer Schrift mit gelben Hintergrund angezeigt

```
$ ls -l test.sh
-rwxr-sr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

```
$ chmod g+s test.sh
$ ls -l test.sh
-rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
```

Spezielle Berechtigungen - Set UID

- ➔ SUID o. Set User ID
- ➔ oktaler Wert 4 und mit s in den Benutzer-Berechtigungen im Symbolic Mode
- ➔ Ähnliche Wirkung wie das SGID-Bit, nur dass der Prozess mit den Rechten des Benutzers ausgeführt wird, dem die Datei gehört.
- ➔ Ein s anstelle des x in den Berechtigungen für den Benutzer
- ➔ Bei farblichen Terminal: Verzeichnis wird in schwarzer Schrift mit rotem Hintergrund angezeigt

```
-rwsr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

```
$ chmod 6755 test.sh
```

```
$ ls -lh test.sh
-rwsr-sr-x 1 carol carol 66 Jan 18 17:29 test.sh
```

Aufgaben

1. Erstelle ein Verzeichnis namens emptydir mit dem Befehl mkdir emptydir. Liste nun mit ls die Berechtigungen für das Verzeichnis emptydir auf.
2. Erstelle eine leere Datei namens emptyfile mit touch. Nun füge mit chmod in symbolischer Schreibweise Ausführungsrechte für den Benutzer emptyfile hinzu und entferne die Schreib- und Ausführungsrechte für alle anderen.
3. Wie lauten die Berechtigungen für eine Datei namens text.txt, nachdem man den Befehl chmod 754 text.txt benutzt hat?

Aufgaben

1. Angenommen eine Datei namens test.sh ist ein Shell-Skript mit den folgenden Berechtigungen und Eigentumsverhältnissen:

```
-rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
```

   a. Was sind die Berechtigungen für den Eigentümer der Datei?
   b. Wenn der Benutzer john dieses Skript ausführt, unter welchen Rechten wird es ausgeführt?
   c. Wie lautet unter Verwendung der numerischen Notation die Syntax von chmod, um die spezielle Erlaubnis, die dieser Datei gewährt wurde, “aufzuheben”?

Aufgaben

1. Betrachte die 4 Dateien und notiere die entsprechenden Berechtigungen für jede Datei und Verzeichnis in numerischer 4-stelliger Schreibweise.

```
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
```

```
drwxr-sr-x 2 carol users 4,0K Jan 18 17:26 Sample_Directory
```

Sicherheitsgrundlagen in Linux

Besondere Dateien und Verzeichnisse

Temporäre Dateien

- ➔ Unter Linux wird eigentlich alles wie eine Datei behandelt
- ➔ Einige Dateien erhalten aber besondere Behandlung
  - ◆ Wegen des Ortes, an dem sie gespeichert sind, wie z.B. temporäre Dateien
  - ◆ Oder wegen der Art und Weise, wie sie mit dem Dateisystem interagieren, wie z.B. Links
- ➔ Temporäre Dateien = werden vom Programm verwendet, um Daten zu speichern, die nur für kurze Zeit benötigt werden.
  - ◆ z.B. Daten laufender Prozesse, Crash-Protokolle, Scratch-Dateien von einem Autosave, Zwischendateien für z.B. eine Dateikonvertierung, Cache-Dateien usw.
- ➔ Nach Version 3.0 des Filesystem Hierarchy Standard (FHS) Konventionen/Empfehlungen:
  - ◆ /tmp = Programme sollen hier nicht davon ausgehen, dass hier abgelegte Dateien zwischen den Aufrufen eines Programms erhalten bleiben. Empfehlung: Verzeichnis während des Systemstarts löschen
  - ◆ /var/tmp = wird während des Systemstarts nicht gelöscht, d.h. hier gespeicherte Daten bleiben normalerweise zwischen den Neustarts erhalten
  - ◆ /run = enthält Daten von Laufvariablen, die von laufenden Prozessen verwendet werden z.B. Prozesskennungsdateien (.pid). Dieser Ort muss während des Systemstarts gelöscht werden. Früher /var/run, manchmal noch ein symbolischer Link zu /run

Berechtigungen für temporäre Dateien

- ➔ Herausforderung bzgl. der Zugriffsrechte
- ➔ Man könnte denken solche Verzeichnisse seien “welt-schreibbar”, d.h. Jeder Benutzer könne darin Daten schreiben oder löschen.. Aber wenn das so wäre, wie könnte man verhindern, dass ein Benutzer Dateien, die von einem anderen erstellt wurden, löscht oder verändert?
- ➔ Lösung → Sticky Bit verhindert, dass Benutzer eine Datei innerhalb eines Verzeichnisses nicht entfernen oder umbenennen können, wenn sie die Datei nicht besitzen

```
$ ls -ldh /tmp/ /var/tmp/
drwxrwxrwt 25 root root 4,0K Jun  7 18:52 /tmp/
drwxrwxrwt 16 root root 4,0K Jun  7 09:15 /var/tmp/
```

Links (Symbolischer vs. Hard Link)

Symbolic Link	Hard Links
- Softlinks - Zeigen auf den Pfad einer anderen Datei - Wenn man die Datei löscht, auf die der Link zeigt (Target bzw. Ziel), existiert der Link zwar noch, aber er “funktioniert” nicht mehr, da er nun auf “nichts” zeigt -	- Hard Link ist ein zweiter Name für die Originaldatei - Er ist kein Duplikat, sondern ein zusätzlicher Eintrag im Dateisystem, der auf dieselbe Stelle auf der Platte (Inode) zeigt

Mit Hard Links arbeiten

```
$ ln target.txt /home/carol/Documents/hardlink
```

- ➔ Mit ln TARGET LINK_NAME erstellen wir einen Hard Linkk
- ➔ Das TARGET muss bereits existieren
- ➔ Besser ist Angabe des absoluten Pfades
- ➔ Wenn man LINK_NAME leer lässt, wird ein Link mit demselben Namen wie das Ziel im aktuellen Verzeichnis erstellt

Mit Hard Links arbeiten

- ➔ Hard Links sind Einträge im Dateisystem, die unterschiedliche Namen haben, aber auf dieselben Daten auf der Festplatte zeigen.
- ➔ Wird der Inhalt an einer Stelle verändert, ändert sich der Inhalt auch an allen anderen Stellen. Wenn eine Stelle gelöscht wird, existieren die anderen dennoch.
  - ◆ Es wird nur die Eintrag in der Dateisystemtabelle gelöscht, der auf den Inode zeigt
- ➔ Überprüfen mit ls -li
- ➔ Hard Links nur auf Dateien (Soft Links auch auf Verzeichnisse)
- ➔ Mit rm und mv löschen bzw. umbenennen

```
$ ls -li
total 224
3806696 -r--r--r-- 2 carol carol 111702 Jun  7 10:13 hardlink
3806696 -r--r--r-- 2 carol carol 111702 Jun  7 10:13 target.txt
```

- Zahl vor der Berechtigung ist die Inode-Nummer
- Hier ist der eine ein Hard Link des anderen (welches Original ist, ist egal)
- Jeder Hard Link erhöht den Link Count
  - Standard: Datei 1 und Verzeichnis 2)

Mit Soft Links arbeiten

- ln -s TARGET LINK_NAME
- Zeigen auf einen anderen Pfad im Dateisystem.
- Für Dateien und Verzeichnisse und auf verschiedenen Partitionen
- Auch wenn rwx für alle angezeigt wird, gelten die Zugriffsrechte von dem Ziel
- Beim rm und mv beachten, dass der Speicherort des Ziels relativ zum Speicherort des Links interpretiert wird wenn der Link oder die Datei, auf die er zeigt, nicht vollständig angegeben wird.

```
$ ls -lh
total 112K
-rw-r--r-- 1 carol carol 110K Jun  7 10:13 target.txt
lrwxrwxrwx 1 carol carol   12 Jun  7 10:14 softlink -> target.txt
```

```
$ ln -s original.txt softlink
```

```
$ mv softlink ../
$ less ../softlink
../softlink: No such file or directory
```

```
$ ln -s /home/carol/Documents/original.txt softlink
```