3.6 Grundkonfiguration/Benutzer und Gruppen

3.6.1 Vorgehensweisen

Zur Benutzerverwaltung sind verschiedene Vorgehensweisen üblich:

a) Editieren der Systemdateien wie `/etc/passwd` von Hand

b) Benutzen von Programmen wie `useradd`, um genau eine Aktion mit einem Schritt zu erledigen

c) Benutzen darauf aufgebauter Skripte wie `adduser`, um zu genau einer Aktion (Benutzer anlegen) dazugehörige Schritte zu erledigen (persönliches Verzeichnis des Benutzers anlegen mit den entsprechenden Berechtigungen); in diese Kategorie gehören auch die meisten graphischen Werkzeuge

d) Benutzen von Serienskripten, um im Stapelbetrieb beliebig viele Aktionen auf einmal abzuarbeiten

3.6.2 Werkzeuge

Hat man sich für eine Vorgehensweise entschieden, muss man die entsprechenden Werkzeuge kennenlernen:

a) Zum Editieren der Systemdateien wie `/etc/passwd` von Hand eignen sich die Programme

- `vipw` – editiert `/etc/passwd` und `/etc/shadow`
- `vigr` – editiert `/etc/gpasswd` und `/etc/gshadow`

b) Auf den meisten Systemen befinden sich folgende Programme, die genau eine Aktion ausführen:

- `useradd` – fügt einen Benutzer hinzu
- `usermod` – ändert Eigenschaften eines Benutzers (auch Gruppenzugehörigkeit)
- `userdel` – löscht einen Benutzer
- `passwd` – ändert das Passwort eines Benutzers
- `groupadd` – fügt eine Gruppe hinzu
- `groupmod` – ändert Eigenschaften eines Benutzers (Namen, gid, Passwort)
- `groupdel` – löscht eine Gruppe
- `gpasswd` – ändert das Passwort einer Gruppe
- `gpasswd` – ändert das Passwort, liest von Datei oder Tastatur

c) Bei Debian-artigen Systemen sind folgende Skripte verfügbar, die einen (etwas) komfortablen Wrapper um die vorgenannten Programme bieten:

- `adduser` – fügt einen Benutzer hinzu nach Vorgaben aus der Datei `adduser.conf`
- `deluser` – löscht einen Benutzer nach Vorgaben aus der Datei `deluser.conf`
- `addgroup` – fügt eine Gruppe hinzu nach Vorgaben aus der Datei `adduser.conf` (!)
- `delgroup` – löscht eine Gruppe nach Vorgaben aus der Datei `deluser.conf` (!)

d) Für das Anlegen vieler Benutzer gibt es vor allem das Programm

- `newusers` – liest zeilenweise aus einer Datei Namen, Passwörter und sonstige Eigenschaften gewünschter neuer Benutzer und legt sie an.

3.6.3 Passwort-Problem

Manche der genannten Werkzeuge erlauben die Angabe eines verschlüsselten Passworts in der Kommandozeile. Das ist zwar nicht empfehlenswert, wird aber trotzdem manchmal gebraucht. Nun gibt es zwar in einer C-Bibliothek die entsprechende Funktion `crypt()`, mit der man das Passwort verschlüsseln kann, der zugehörige Konsolen-Befehl fehlt jedoch leider. Hier gibt es zwei Abhilfe-Möglichkeiten. Entweder man baut sich diesen Befehl selbst aus dem folgenden C-Programm `mycrypt.c`

```
1  #include <crypt.h>
2  #include <stdio.h>
3  #include <unistd.h>
4
5  int main(int argc, char* argv[])
6  {
7      if(argc<3)
8      {
9          fprintf(stderr, "Aufruf: %s password salt,\n"
10                         "   Das 'salt' ist eine beliebige Zeichenkette\n"
11                         "   mit zwei Zeichen, z.B. '42'\n", *argv);
12         return 1;
13     }
14     puts(crypt(argv[1], argv[2]));
15     return 0;
16 }
```

und dem Shell-Befehl

Terminal

```
root@debian964:~# gcc -o mycrypt.c mycrypt.c -lcrypt
```

zusammen, oder man benutzt den Befehl `mkpasswd` aus dem Paket `whois` (!).

3.6.4 Benutzer absichern

3.6.4.1 Prozesse ohne Ende: Die Forkbombe Ein böswilliger Benutzer kann ein System dadurch stören, dass er möglichst viele System-Ressourcen an sich zieht. Andere Benutzer können dann nicht auf das System zugreifen (*denial of service*, DOS-Angriff). Ein Beispiel dafür ist die so genannte Forkbombe (*fork bomb*), bei der er möglichst viele Prozesse nacheinander startet:

Terminal

```
schueler@debian964:~$ gedit forkbomb.c
```

```
1  #include <unistd.h>
2  int main(void)
3  {
4      while(1)
5          fork();
6  }
```

Terminal

```
schueler@debian964:~$ make forkbomb
schueler@debian964:~$ forkbomb
bash: fork: Die Ressource ist zur Zeit nicht verfügbar
 Strg + C
```

Bei der Forkbombe findet durch die Verdoppelung des Prozesses eine exponentielle Zunahme der Prozesse auf dem System statt. Irgendwann können keine neuen Prozesse mehr angelegt werden.

3.6.4.2 Grenzen setzen mit dem Befehl ulimit Damit das nicht passieren kann, gibt es den Shell-Befehl `ulimit`, mit dem man Maximalwerte für bestimmte Ressourcen setzen (und anzeigen) kann. Die Option `-a` dient zur Anzeige aller momentanen Werte:

Terminal
```
schueler@debian964:~$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 31822
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 1000
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

Hier sind maximal 1000 gleichzeitig laufende Prozesse für jeden Benutzer möglich. Nun kann man diesen Wert herabsetzen:

Terminal
```
schueler@debian964:~$ ulimit -u 700 # setzen
schueler@debian964:~$ ulimit -u     # ansehen
700
```

Dieser Wert ist allerdings nur eine Warngrenze (*soft limit*): Bei mehr als 700 Prozessen bekommt der aktuelle Nutzer einen Warnhinweis. Mit der zusätzlichen Option `-H` kann man eine feste Grenze einbauen (*hard limit*), die nicht überschritten werden kann:

Terminal
```
schueler@debian964:~$ ulimit -Hu 800 # setzen
schueler@debian964:~$ ulimit -Hu     # ansehen
800
```

Der Befehl `ulimit` veranlasst einen Systemaufruf, der dem aktuellen Prozess die Eigenschaft verleiht, nur noch die begrenzte Ressource zu bekommen. Diese Eigenschaft kann nicht zurückgenommen werden. Diese Eigenschaft wird auch an alle Kindprozesse und weitere Nachkommen vererbt. Daher eignet sich `ulimit` sehr gut dafür, ein System gegen Forkbomben und ähnliche Störversuche abzusichern. Dazu müsste man `ulimit`-Befehle in die Dateien `/etc/profile` (Initialisierung von Login-Shells) und `/etc/bash.bashrc` (Initialisierung von interaktiven Shells) schreiben.

3.6.4.3 Grenzen setzen mit der Datei limits.conf In Debian gibt es die Datei `limits.conf` im Verzeichnis `/etc/scecurity`. Hiermit kann man die `ulimit`-Befehle für alle gewünschten Benutzer einfach einbinden:

```
1 #<domain> <type> <item> <value>
2 meier      soft   nproc  500
3 meier      hard   nproc  600
4 @ifs9a     soft   nproc  400
5 @ifs9a     hard   nproc  420
```

Mit dem Zeichen wird eine Gruppe gekennzeichnet, mit $*$ sind alle Benutzer gemeint. Weitere Einzelheiten findet man in der Datei `limits.conf` selbst und in der Manualpage dazu.

Die Belegung von Massenspeichern wird durch `limits.conf` nicht begrenzt. Für dieses Problem gibt es das Paket `quota` (siehe dort).

3.6.5 Weitere Konfiguration

Oft sollen weitere Systemeigenschaften für Benutzer angepasst werden:

- Default-Rechte für von den Benutzern neu erzeugte Dateien und Verzeichnisse – dazu gibt es das Programm `umask`. Auch hier kann man einen Aufruf in die Dateien `/etc/profile` und `/etc/bash.bashrc` bringen. Anders als bei `ulimit` ist dieser Wert nicht sicherheitskritisch, und der Benutzer kann ihn beliebig ändern.
- Aliase und Suchpfadlisten sind am besten in der Datei `.bashrc` im persönlichen Verzeichnis des Benutzers aufgehoben. Dort kann er sie jederzeit anpassen.