Kapitel 3: Zugriffskontrolle

3: Zugriffskontrolle

Zugriffskontrolle

Satz von Regeln die festlegen welche Subjekte mit welchen Rechten auf welche Objekte zugreifen dürfen.

Realisierungsansätze

- Whitelist
  - Regelsatz der definiert was erlaubt ist.
  - Verboten ist alles, was nicht explizit erlaubt ist.
- Blacklist:
  - Regelsatz der definiert was verboten ist.
  - Erlaubt ist alles, was nicht explizit verboten ist.
- Hybride Listen.

3.1: Allgemeine Zugriffskontrolle

Wir betrachten drei Mengen,

1. Eine Menge S von Subjekten,
2. Eine Menge O von Objekten und
3. Eine Menge R von Rechten (=Zugriffsarten).

Zugriffskontrollmatrix

Eine Zugriffskontrollmatrix A ist eine zweidimensionale $|S| \times |O|$ Whitelist.

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1|O|} \\ a_{21} & a_{22} & \dots & a_{2|O|} \\ \vdots & \vdots & \vdots & \vdots \\ a_{|S|1} & a_{22} & \dots & a_{|S||O|} \end{pmatrix}$$

Bei den einzelnen Einträgen a_{ij} handelt es sich um Teilmengen von R, d.h. es gilt: $a_{ij} \in R$.

Beispiel: Zugriffskontrollmatrix

- $S = \{\text{Alice, Bob, Gast}\}$
- $O = \{\text{Alice-Tagebuch, Einkaufsliste, Klingel }\}$
- $R = \{\text{lesen, schreiben }\}$

Aufgabe: Erstellen Sie eine sinnvolle Zugriffskontrollmatrix.

Spezielle Darstellungsarten

Zugriffskontroll-Listen (“Access Control List”, ACL)

Zu jedem Objekt $o \in O$ existiert eine Liste von Paaren $(s, r) \in S \times R$.

Capability-Listen

Zu jedem Subjekt $s \in S$ existiert eine Liste von Paaren $(o, r) \in O \times R$, oder zu jedem $(s, r) \in S \times R$ existiert eine Liste von Objekten $o \in O$.

Beispiel: ACL und Capability-Liste

- $S = \{$Alice, Bob, Gast$\}$
- $O = \{$Alice-Tagebuch, Einkaufsliste, Klingel $\}$
- $R = \{$lesen, schreiben $\}$

Aufgabe: Erstellen Sie eine sinnvolle ACL und Capability-Liste.

3.2: Zugriffskontrolle in UNIX (Fallbeispiel)

- Jedes *Objekt* (Datei, Verzeichnis, Socket, Symbolischer Link, Gerät (Block oder Character), FIFO) ist **genau einem Eigentümer**(-*Subjekt*) und **genau einer Gruppe** (von *Subjekten*) zugeordnet.
- Für den Eigentümer, für die Gruppe und für alle anderen Benutzer (die "Welt") kann man jeweils getrennt festlegen, ob sie die *Methoden* **r**ead, **w**rite und e**x**ecute ausführen dürfen.

Beispiel:

```
$ ls -lah sysprog03.pdf
-rw-r--r-- 1 cforler dozent 161K Mär 16 14:50 sysprog03.pdf
```

Zugriffsrechte Verstehen

```
$ ls -lah sysprog03.pdf
-rw-r--r-- 1 cforler dozent 161K Mär 16 14:50 sysprog03.pdf
```

Bedeutung der Spaltenbasierten-Ausgabe von `ls`

- 1. Spalte: Typ und Zugriffsrechte
  - Das erste Zeichen steht für den Dateitypen (`-`: Datei; `b`: Block device; `c`: Character device; `d`: Verzeichnis; `l` symbolicher Link; `p`: named pipe (FIFO); `s`: Socket)
  - Die nächsten drei Zeichen: Zugriffsrechte des Eigentümers (u)
  - Die nächsten drei Zeichen: Zugriffsrechte der Gruppe (g)
  - Die nächsten drei Zeichen: Zugriffsrechte aller Anderen (o)
- 2. Spalte Anzahl Links
- 3. und 4. Spalte: Name und Gruppe des Eigentümers
- 5. Spalte: Dateigröße
- 6-8. Spalte: Zeitstempel des letzten Schreibzugriffes
- 9. Spalte Dateiname

Zugriffsrechte ändern mittels `chmod`

Mit dem Tool `chmod` lassen sich die Zugriffsrechte ändern

Allen Schreibrechte entziehen

```
$ chmod -w sysprog03.pdf
$ ls -lah sysprog03.pdf
-r--r--r-- 1 cforler cforler 162K Mär 16 14:55 sysprog03.pdf
```

Den Anderen Leserechte entziehen

```
$ chmod o-r sysprog03.pdf
$ ls -lah sysprog03.pdf
-r--r----- 1 cforler cforler 163K Mär 16 14:57 sysprog03.pdf
```

Der Gruppe Schreibrechte erteilen

```
$ chmod g+w sysprog03.pdf
$ ls -lah sysprog03.pdf
-r--rw---- 1 cforler cforler 163K Mär 16 14:57 sysprog03.pdf
```

Zugriffsrechte als dreistellige Oktalzahl

- Jedem Zugriffsrecht wird eine Zahl (Zweierpotenz) zugeordnet
  - **Lesen (read): 4**
  - **Schreiben (write): 2**
  - **Ausführen (execute): 1**
- Jede Ziffer der Oktalzahl ist die Summe der Rechte.
- Erste Ziffer: Rechte des Eigentümers.
- Zweite Ziffer: Rechte der Gruppe.
- Dritte Ziffer: Rechte der Anderen.

```
$ chmod 660 sysprog03.pdf
```

Frage: Was bewirkt das obige Kommado?

Execute or Write

- In der Regel sollte man entweder Rechte zum ausführen
- oder Rechte zum schreiben verteilen.
- Achtung: Lesen erlaubt das kopieren von Dateien.

Merke: Lesen ermöglicht ausführen

- Lesen erlaubt kopieren.
- Kopierer ist Eigentümer der kopierten Datei.
- Kopierer kann kopierte Datei die Rechte zum ausführen geben.
- Das ist uns nicht 1337 genug. Es geht noch besser. :-)
- Dynamisch gelinkte Programme können auch direkt von einem dynamischen *Loader* (`ld.so`) ausgeführt werden.
- Bsp: `$ /lib64/ld-linux-x86-64.so.2 /bin/ls`

Das SUID Bit

- Normalerweise läuft ein Programm mit den Rechten des Aufrufers.
- Der Eigentümer kann eine ausführbaren Datei `foo` mittels `chmod u+s foo` als `suid` markieren (“set user id”).
- Ist das `suid`-Bit gesetzt, dann läuft das Programm mit den Rechten des **Eigentümers**.

  Analog: `sgid` (“set group id”).

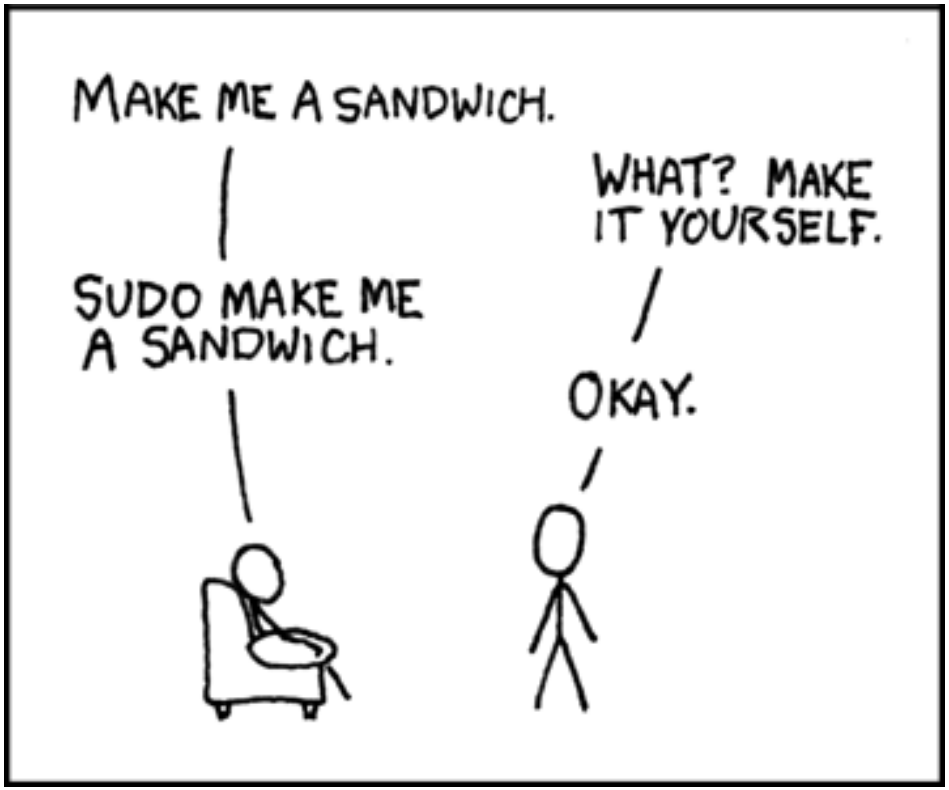
Sudo

- Das Programm `sudo` ermöglicht es regulären Benutzer Programm als `root` auszuführen.
- `root` ist unter UNIX einen Super-User Account (`root`) dessen Zugriffsrechte praktisch unbeschränkt sind (unter Windows entspricht dies dem Account `system`).

Beispiel:

```
$ whoami
cforler
$ ls -lah /usr/bin/sudo
-rwsr-xr-x 1 root root 126K Feb 23  2015 /usr/bin/sudo
$ sudo whoami
root
```

Das Allgegenwärtige Sudo



Quelle: `https://xkcd.com/149/`

Anmerkungen

- Natürlich gibt es UNIX-Varianten, leistungsfähigere Zugriffskontrollen wie ACLs.
- Das `suid`/`guid` Verfahren erlaubt es, präzise Zugriffsstrukturen festzulegen. Ein Benutzer *s* kann sogar das eigene Recht, eine Methode auf ein Objekt *o* anwenden zu können, auch anderen gewähren – ohne dass *s* selbst Eigentümer von *o* ist.
- Leider ist diese Verfahren auch kompliziert und fehlerträchtig. In der Praxis neigen viele Programmierer dazu, ausführbaren Dateien zu weitreichende Privilegien einzuräumen ("`suid root` – alles wird gut" :`-)` . Dies ist die Ursache für viele UNIX-Sicherheitslücken.
- Kompliziertheit und Sicherheit sind natürliche Feinde!

3.3: Zugriffskontrollstrategien

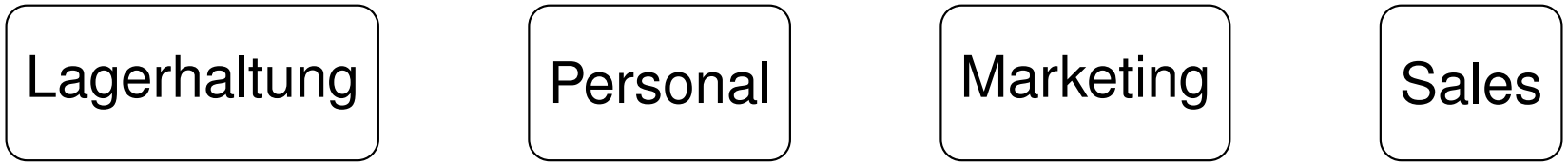
Bis jetzt hatten wir uns “Zugriffskontrolltaktiken” angesehen mit denen wir die Vertraulichkeit von Daten schützen können.

Es ging darum, wie man *Subjekten* erlaubt oder verbietet, mit bestimmten *Methoden* auf *Objekte* **zuzugreifen**.

Nach welchen Gesichtspunkten wird dies überhaupt festgelegt?

Darum geht es in dem Rest dieses Kapitels.

Zugriffskontrollstrategie: Abteilungen (1/2)

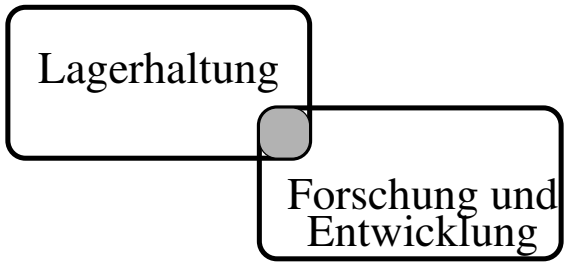


- Abteilungen (Compartments) sind Gruppen, die immer *Subjekte* und *Objekte* enthalten.
- Jedes Subjekt hat Zugriff auf Objekte seiner eigenen Gruppe.
- Keine Hierarchie oder sonstige Relation zwischen den Gruppen.

Zugriffskontrollstrategie: Abteilungen (2/2)

Erweiterung:

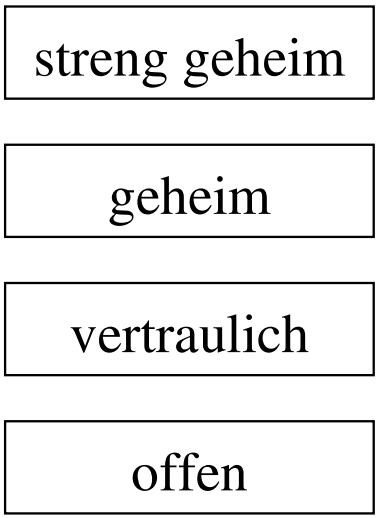
Subjekte und *Objekte* dürfen mehreren Gruppen angehören.



Typische Regel der Zugriffskontrolle:

Ein Subjekt *s* darf auf ein Objekt *o* zugreifen, wenn *s* (mindestens) **allen Gruppen** angehört, zu denen auch *o* gehört.

Zugriffskontrollstrategie: Sicherheitsstufen



- Jedes *Subjekt* gehört zu genau einer Gruppe (“clearance”). Jedes *Objekt* gehört zu genau einer Gruppe (“classification”).
- Die Gruppen sind linear geordnet (“*. . . ist weniger kritisch als . . .* ”).

Regeln für Subjekt s und Objekt o:

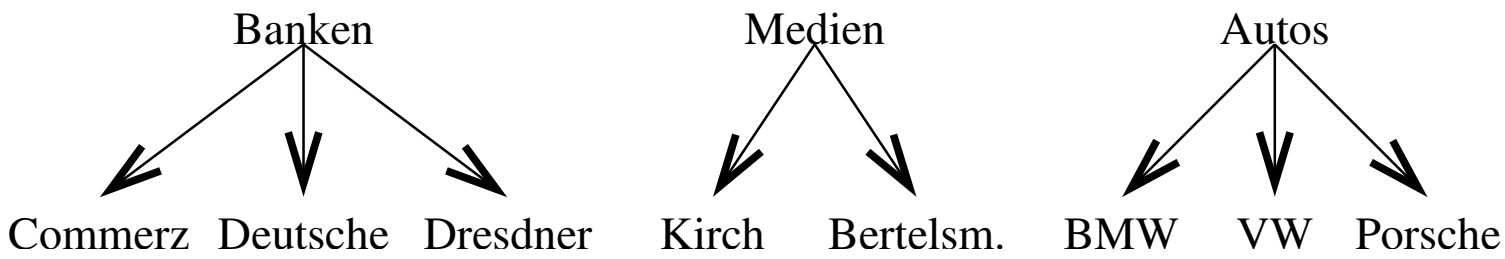
- Ist $s \geq o$, darf s **lesend** auf o zugreifen.
- Ist $s \leq o$, darf s **schreibend** auf o zugreifen.

Frage: Weshalb sind die obigen Regeln sinnvoll?

Zugriffskontrollstrategie: Chinesische Mauer (1/3)

- Sehr spezielle kommerzielle Strategie.
- *Zweistufige Hierarchie* von (Gruppen von) Objekten: Branchen und Firmen.

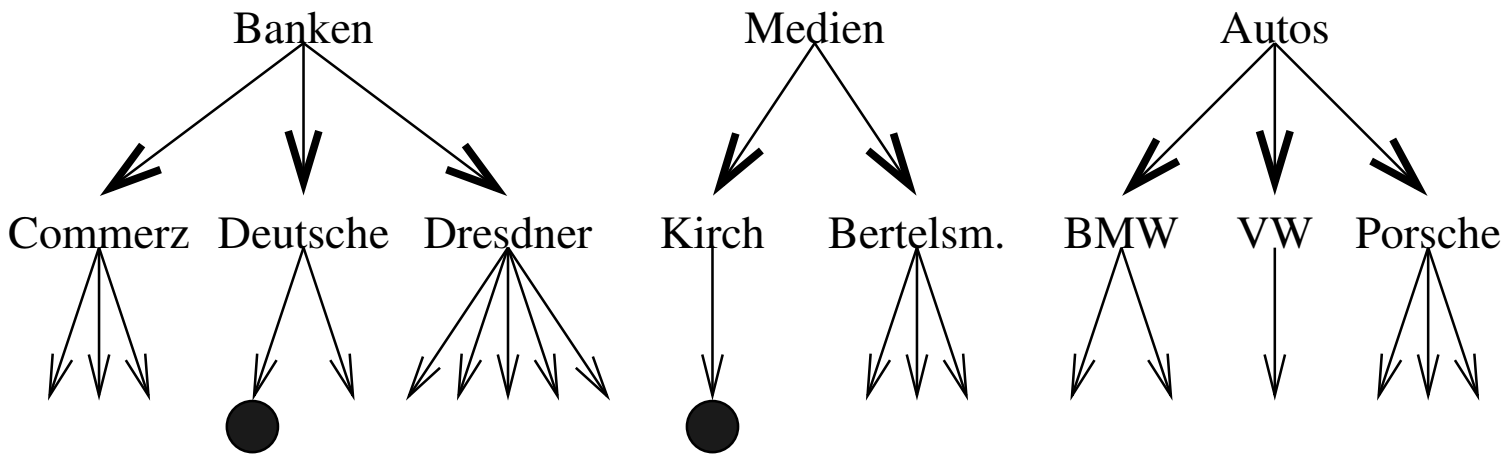
Beispiel:



Zugriffskontrollstrategie: Chinesische Mauer (2/3)

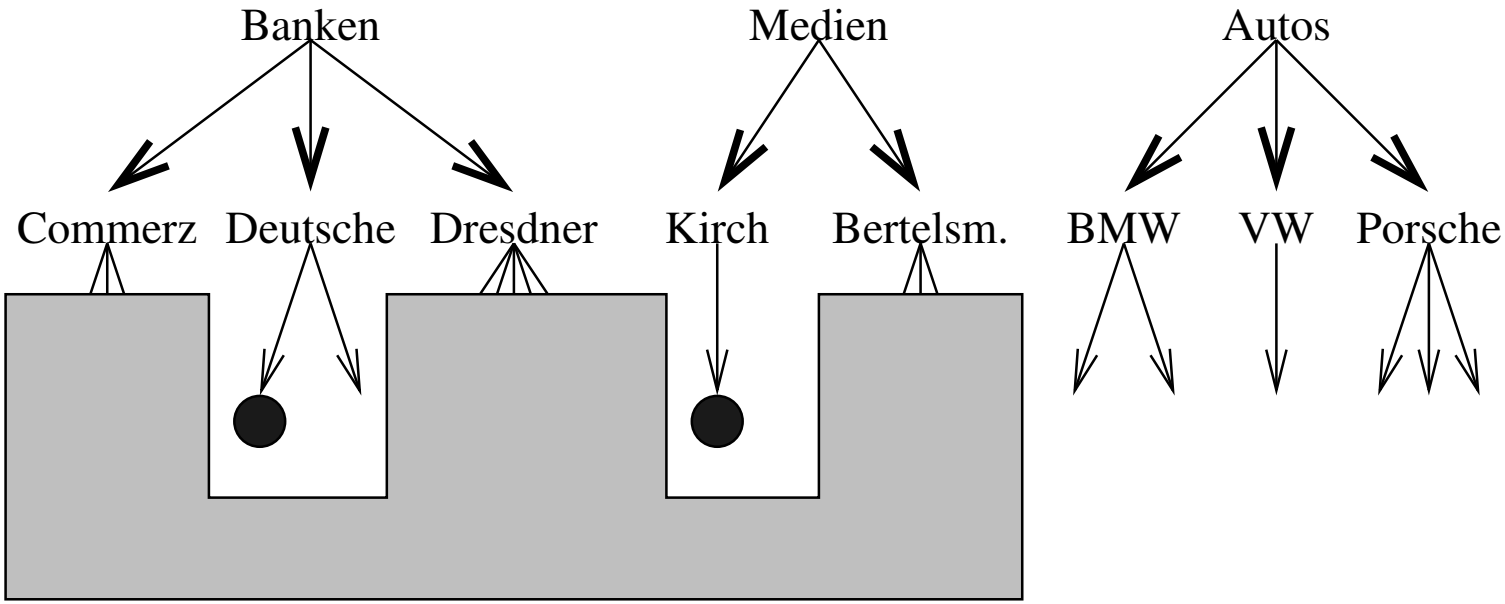
- *Subjekte* sind *Personen* (“Berater”).
- Regel: Wer auf Objekte einer Firma F in einer Branche B zugreift, darf nicht auf Objekte einer Firma $F^* \neq F$ der gleichen Branche B zugreifen.

Beispiel: Eine Person greift auf zwei Objekte zu.



Zugriffskontrollstrategie: Chinesische Mauer (3/3)

Nun liegen 11 Objekte “*hinter* der chinesischen Mauer”:



Zusammenfassung

Sie sollten . . .

- . . . wissen was eine Zugriffskontrollmatrix ist.
- . . . wissen was eine Zugriffskontroll-Liste ist.
- . . . wissen was eine Capability-Liste ist.
- . . . verstanden haben wie die Zugriffskontrolle unter UNIX funktioniert.
- . . . die vorgestellten Zugriffskontrollstrategien verinnerlicht haben.