TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Professur für Betriebssysteme

BETRIEBSSYSTEME UND SICHERHEIT

mit Material von Olaf Spinczyk,
Universität Osnabrück

Systemsicherheit

https://tud.de/inf/os/studium/vorlesungen/bs

HORST SCHIRMEIER

Inhalt

- Überblick über Sicherheitsprobleme
- Rechteverwaltung
  - Schutzmatrix
  - *Access Control Lists*
  - *Capabilities*
  - *Mandatory Access Control*
- Systemsoftware und Sicherheit
  - Hardwarebasierter Schutz
  - Softwarebasierter Schutz
- Zusammenfassung

Silberschatz, Kap. …
14: *Protection*
15: *Security*

Tanenbaum, Kap. …
9: IT-Sicherheit

Inhalt

- **Überblick über Sicherheitsprobleme**
- Rechteverwaltung
  - Schutzmatrix
  - *Access Control Lists*
  - *Capabilities*
  - *Mandatory Access Control*
- Systemsoftware und Sicherheit
  - Hardwarebasierter Schutz
  - Softwarebasierter Schutz
- Zusammenfassung

Sicherheitsprobleme

- **Begriffsdefinition „Sicherheit“**
  - ***Safety***
    - Schutz vor Risiken durch (Software-)Fehler, Störungen oder Ausfällen
  - ***Security***
    - Schutz von Menschen und Rechnern vor intendierten Fehlern (Angriffen)
- Beide Themenbereiche sind für Systemsoftware relevant
  - Im Folgenden geht es um Sicherheit im Sinne von *Security*.
- Ausnutzung von Sicherheitslücken
  - Schadsoftware („*Malware*“)
  - *Social Engineering*

Sicherheit in Betriebssystemen

Jemanden ...

- Unterscheidung von **Personen** und **Gruppen von Personen**

davon abhalten ...

- durch technische und organisatorische Maßnahmen

einige ...

- Begrenzung durch unser Vorstellungsvermögen

unerwünschte Dinge zu tun.

- 1) nicht autorisiert Daten lesen (**Geheimhaltung**, **Vertraulichkeit**),
- 2) nicht autorisiert Daten schreiben (**Integrität**),
- 3) unter „falscher Flagge“ arbeiten (**Authentizität**),
- 4) nicht autorisiert Ressourcen verbrauchen (**Verfügbarkeit**),
- usw.

- Unterscheidung zwischen Angriffen von
  - innen
  - außen

Beispiel: Login-Attrappe

- Angreifer startet ein Benutzerprogramm, das am Bildschirm einen Login-Screen simuliert.
- Der ahnungslose Benutzer tippt Benutzername und sein privates Passwort.
  - Angreiferprogramm speichert Benutzername und Passwort in Datei.
  - Angreiferprogramm terminiert aktuelles Shell-Programm

➔ Login-Sitzung des Angreifers wird beendet und regulärer Login-Screen wird angezeigt

- Abhilfe: Login-Sequenz durch Tastensequenz starten, die von Benutzerprogramm nicht abgefangen werden kann
  - z.B. CTRL-ALT-DEL bei Windows NT und folgende.

Beispiel: Virus

- Programm, dessen Code an ein anderes Programm angefügt ist und sich auf diese Weise **reproduziert**
  - Virus schläft, bis infiziertes Programm ausgeführt wird
  - Start des infizierten Programms führt zur Virusreproduktion
  - Ausführung der Virusfunktion ist u.U. zeitgesteuert
- Virusarten
  - **Bootsektor-Virus**: beim Systemstart
  - **Makro-Virus**: in skriptbaren Programmen wie Word, Excel
    - Durch Dokumente verbreitet!
  - Ausführbares Programm als Virus
- Verbreitung durch
  - Austausch von Datenträgern
  - Mail-Attachments
  - Webseiten

Arten von „Schädlingen“ (1)

- **Viren**
  - Durch Anwender unabsichtlich verbreitete Programme
  - schleusen sich in andere Programme ein
  - reproduzieren sich damit
- **Würmer**
  - warten nicht, von Anwender auf neues System verbreitet zu werden
  - versuchen, aktiv in neue Systeme einzudringen
  - Ausnutzung von Sicherheitslücken auf Zielsystemen
- **Trojanische Pferde** („Trojaner“)
  - Programm, das als nützliche Anwendung getarnt ist
  - Zudem wird ohne Wissen des Anwenders andere Funktion erfüllt, z.B. Netzwerkzugang für den Angreifer

Arten von „Schädlingen“ (2)

- ***Rootkit***
  - Sammlung von Softwarewerkzeugen, um …
    - zukünftige Logins des Eindringlings verbergen
    - Prozesse und Dateien zu verstecken
  - wird nach Einbruch in ein Computersystem auf dem kompromittierten System installiert
  - Versteckt sich selbst und seine Aktivitäten vor dem Benutzer
    - z.B. durch Manipulation der Werkzeuge zum Anzeigen von Prozessen (ps), Verzeichnisinhalten (ls), Netzwerkverbindungen (netstat) …
    - … oder durch Manipulation von systemweiten *shared libraries* (libc)
    - … oder direkt durch Manipulation des Systemkerns

- Oft treten Schädlinge als **Kombination** der vorgestellten Kategorien auf.

Inhalt

- Überblick über Sicherheitsprobleme
- **Rechteverwaltung**
  - Schutzmatrix
  - *Access Control Lists*
  - *Capabilities*
  - *Mandatory Access Control*
- Systemsoftware und Sicherheit
  - Hardwarebasierter Schutz
  - Softwarebasierter Schutz
- Zusammenfassung

Ziele

- Schutz gespeicherter Information vor
  - Verletzung der Vertraulichkeit
  - Diebstahl
  - unerwünschter Manipulation (einschl. Verschlüsselung, ***Ransomware***)
- in allen Mehrbenutzersystemen
  - … und jedes am Internet angeschlossene System ist *de facto* ein Mehrbenutzersystem!

Anforderungen

- alle **Objekte** eines Systems müssen eindeutig und fälschungssicher identifiziert werden
- (externer) **Benutzer** eines Systems muss eindeutig und fälschungssicher identifiziert werden
  - ➔ Authentifizierung
- Zugriff auf Objekte nur, wenn Benutzer nötige **Rechte** hat

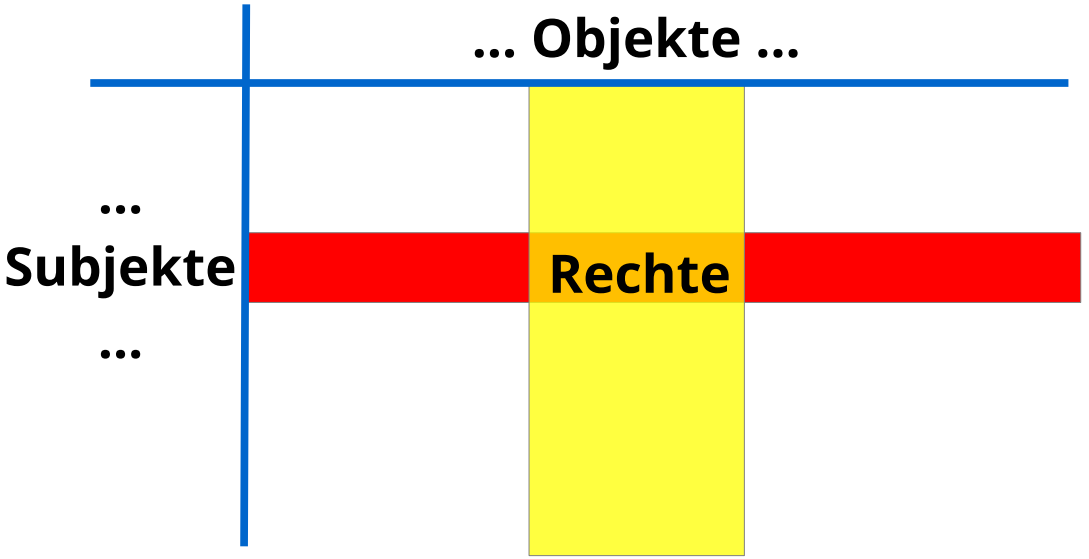
- Zugriff auf Objekte sollte nur über zugehörige **Objektverwaltung** geschehen
  - Rechte müssen **fälschungssicher** gespeichert werden; Weitergabe von Rechten darf nur **kontrolliert** erfolgen
  - grundlegenden Schutzmechanismen sollen **ohne großen Aufwand** überprüft werden können.

Entwurfsprinzipien

- **Prinzip der geringst-möglichen Privilegisierung**
  („*principle of least privilege*“)
  - Person oder Software-Komponenten nur die Rechte einräumen, die für die zu erbringende Funktionalität erforderlich sind
  - **Verbot als Normalfall**
  - Gegenbeispiel: Unix „root“
- **Sichere Standardeinstellungen** („*fail-safe defaults*“)
  - Beispiel: neu installierte Server-Software
- **Separierung von Privilegien** („*separation of duties*“)
  - mehrfache Bedingungen für die Zulassung einer Operation

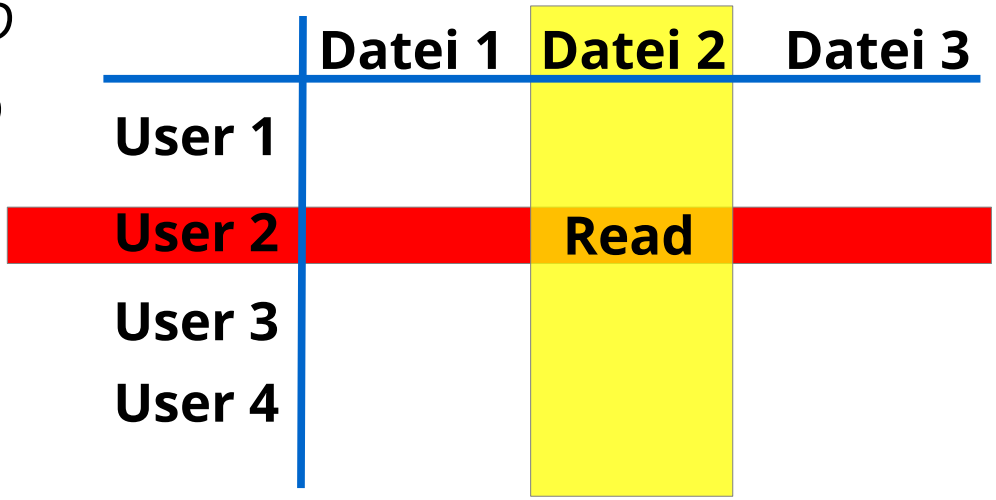
Zugriffsmatrix

- Begriffe:
  - **Subjekte** (Benutzer, Personen, Prozesse)
  - **Objekte** (Daten, Geräte, Prozesse, Speicher …)
  - **Operationen** (Lesen, Schreiben, Löschen, Ausführen …)
- Frage: Ist Operation(Subjekt, Objekt) zulässig?



Basismodell: Datei-/Prozessattribute

- Festlegungen in Bezug auf Benutzer:
  - Für welchen Benutzer arbeitet ein Prozess?
  - Welchem Benutzer gehört eine Datei (*owner*)?
  - Welche Rechte räumt ein Benutzer anderen und sich selbst an „seiner“ Datei ein?
- Rechte eines Prozesses an einer Datei
  - Attribute von Prozessen: *User ID*
  - Attribute von Dateien: *Owner ID*

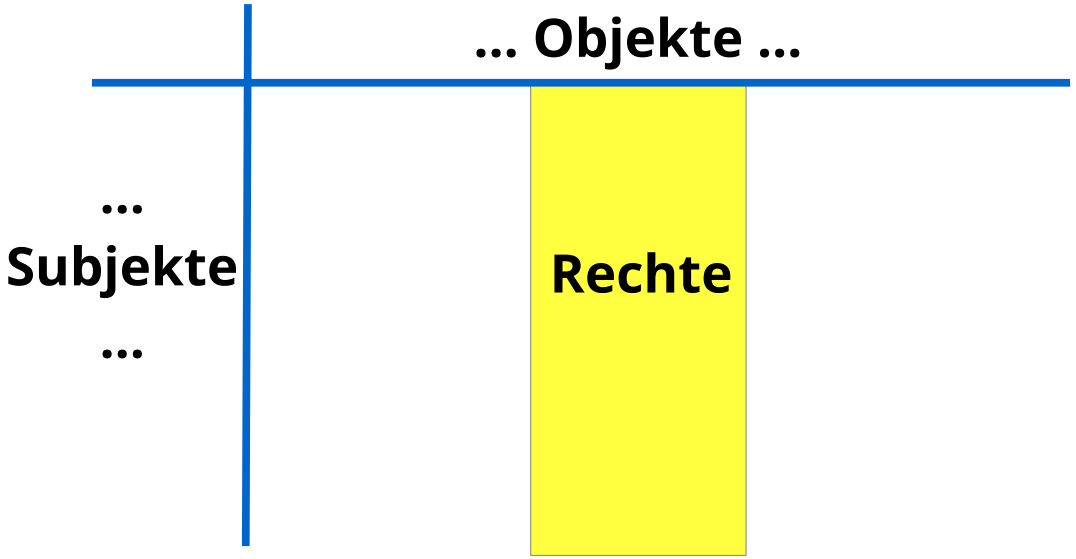


(Implementierungs-)Varianten der Schutzmatrix

- spaltenweise: **ACL – *Access Control List*** (Zugriffssteuerliste)
  - bei jedem Zugriff wird beim Objekt auf der Basis der Identität des Absenders dessen Berechtigung geprüft
- zeilenweise: ***Capabilities*** (Zugriffsberechtigungen)
  - bei jedem Zugriff wird etwas geprüft, was Subjekte **besitzen** und bei Bedarf **weitergeben** können
- regelbasiert: **MAC – *„mandatory access control“***
  - bei jedem Zugriff werden Regeln ausgewertet

ACLs

- Spaltenweise Darstellung: ***Access Control Lists***
  - pro Objekt eine **Liste von Subjekten mit zugehörigen Rechten**
- Analogie: Gästeliste (mit Zusatzinfos: darf Backstage, darf kostenlos an Getränke, …)



Access Control Lists

- Setzen der ACLs darf
  - wer einen ACL-Eintrag für dieses Recht hat
  - Erzeuger der Datei
- Beispiel: Multics – Liste von Tripeln (Nutzer, Gruppe, Rechte)

```
File 0 (jan, *, RWX)
File 1 (jan, system, RWX)
File 2 (jan, *, RW-), (els, staff, R--), (maaike, *, RW-)
File 3 (*, student, R--)
File 4 (jelle, *, ---), (*, student, R--)
```

- Beispiel: Windows (ab NT)
  - Objekt: Vergabe von Rechten (*allow*) und ggf. Ausnahmen (*deny*)
  - Rechte z.B. *full control*, *modify*, *read*, *execute*, ...

Unix-Zugriffsrechte

- Beispiel Unix: rudimentäre Zugriffssteuerlisten
  - Prozess: *User ID*, (eine oder mehrere) *Group IDs*
  - Datei: *Owner*, *Group*
  - Rechte in Bezug auf *User* (*Owner*), *Group*, *Others*

Datei.tex		
`rw-`	`r--`	`---`
		Others
	Group: staff	
User: me		

Dateiattribute:

`rwx`

- Ausführen (*e**X**ecute*): ja/nein
- Schreiben (***W**rite*): ja/nein
- Lesen (***R**ead*): ja/nein

- Neuere Unix-Systeme implementieren auch ACLs
  - siehe **get**/**setfacl**(1)
  - Problem: Dateisystem-Integration, Kompatibilität mit Anwendungen

Problem: Rechte-Erweiterung

- **Beispiel:** Tetris-*Highscore*-Liste
  - gespeichert in: `/home/me/spiele/tetris/Highscores`
  - Programm: `/home/me/bin/spiele/tetris`
  - Anforderung: Jeder soll spielen und sich auch in der *Highscore*-Liste verewigen können.
- Lösung 1: alle haben Schreibrecht
  - ➔ zu viele Rechte (funktioniert nicht)
  - Jeder Benutzer könnte *Highscores* beliebig manipulieren, z.B. Texteditor.
- Lösung 2: *setuid*
  - nur Nutzer „me“ hat Schreibrecht auf *Highscore*-Datei
  - Tetris-Programm mit gesetztem *setuid*-Bit
  - sobald ein Prozess Tetris aufruft, erhält dieser als *User-ID* den *Owner* des ausführbaren Programms

Unix: Benutzer und Prozesse

- Jeder Prozess repräsentiert einen Benutzer
- Prozess-Attribute:
  - *User ID* (**uid**), ≥1 *Group ID* (**gid**)
  - *Effective-uid* (**euid**), *Effective-gid* (**egid**)
    - Bestimmen beim Zugriff auf Dateien die Rechte eines Prozesses
- Nur wenige hoch-privilegierte Prozesse dürfen uid und gid manipulieren
  - z.B. login-Prozess
  - Nach Überprüfung des Passworts setzt Login-Prozess uid, gid, euid, egid.
  - Alle anderen Prozesse: Kinder des Login-Prozess.
- Kind-Prozesse erben Attribute von Eltern

Unix-Lösung: setuid-Mechanismus

- Datei, die vertrauenswürdigen Programmcode (z.B. Tetris) enthält, besitzt Kennzeichnung als *setuid* (**`s`**-Bit)
  - im Verzeichnislisting „s“ statt „x“ für Executable (z.B. `/usr/bin/passwd`)
  - Es gibt auch (seltener verwendet) ein *setgid*-Bit.
- **`exec`** auf *setuid*-Programme
  - ausführender Prozess erhält als effektive UID die UID des Owners des Programms (genauer: der Datei, die das Programm enthält)
  - Prozessausführung erfolgt unter den Rechten dieses Benutzers, solange der Prozess läuft
    - **Problem:** Widerspricht dem Prinzip der geringsten Privilegierung
    - *Workaround*: Speziellen Nutzer für die Applikation einrichten; nicht 'root' verwenden

Beispiel: Highscore-Liste

Shell

uid: fritz
gid: studis
euid: fritz
egid: studis

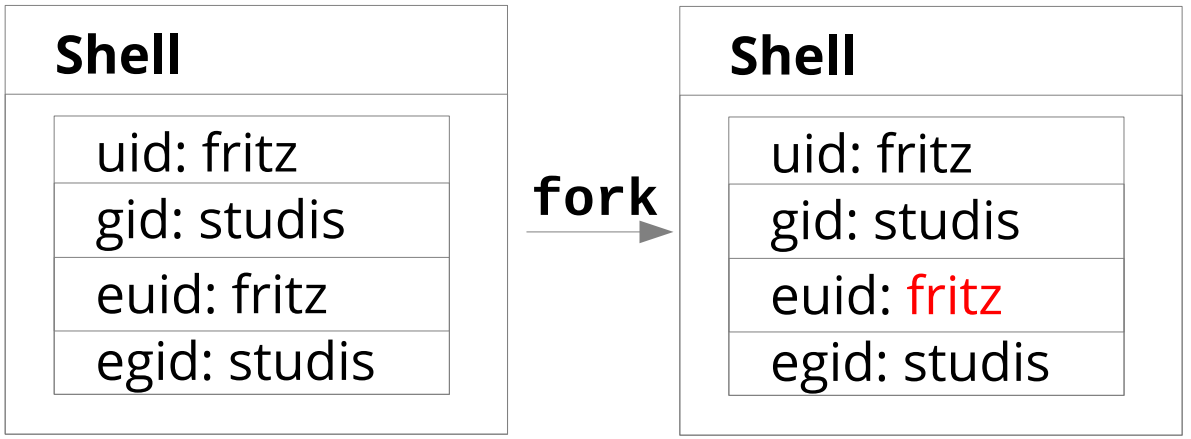
tetris

`r-s`	`--x`	`---`
		Others
	Group: tetris	
User: me		

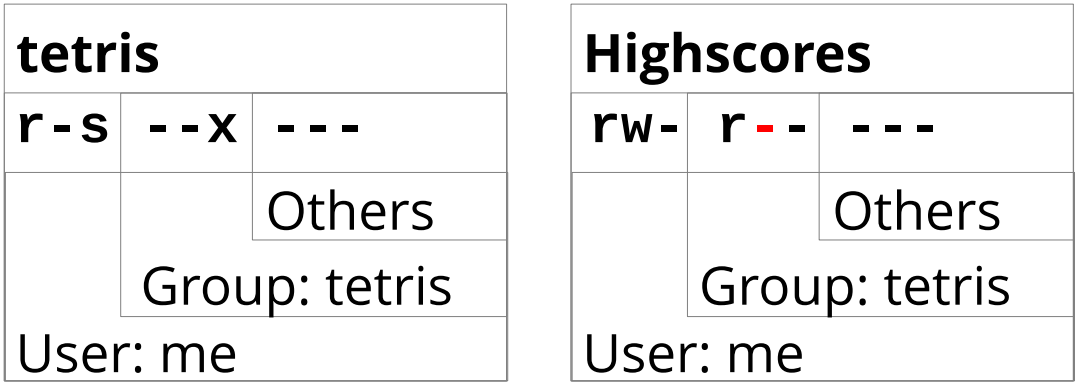
Highscores

`rw-`	`r--`	`---`
		Others
	Group: tetris	
User: me		

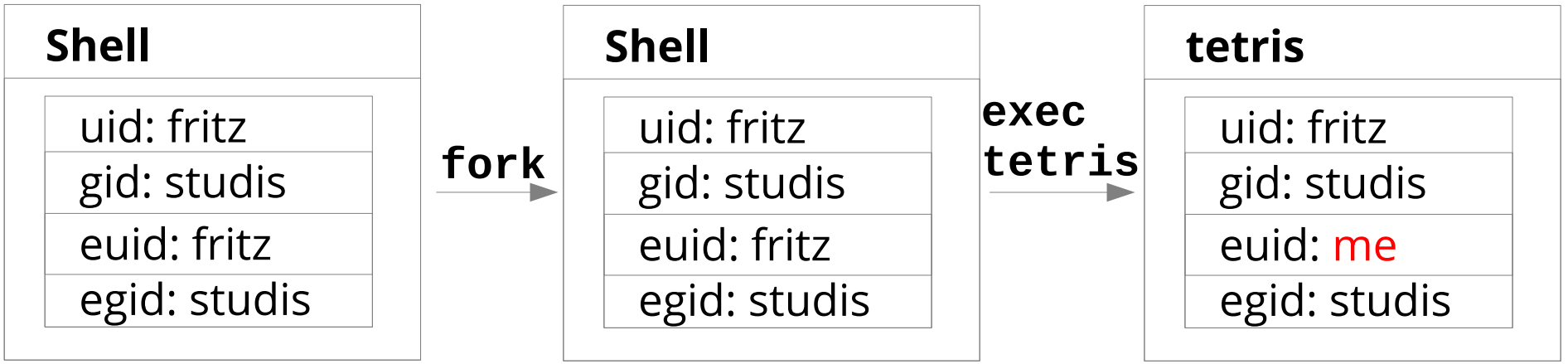
Beispiel: Highscore-Liste



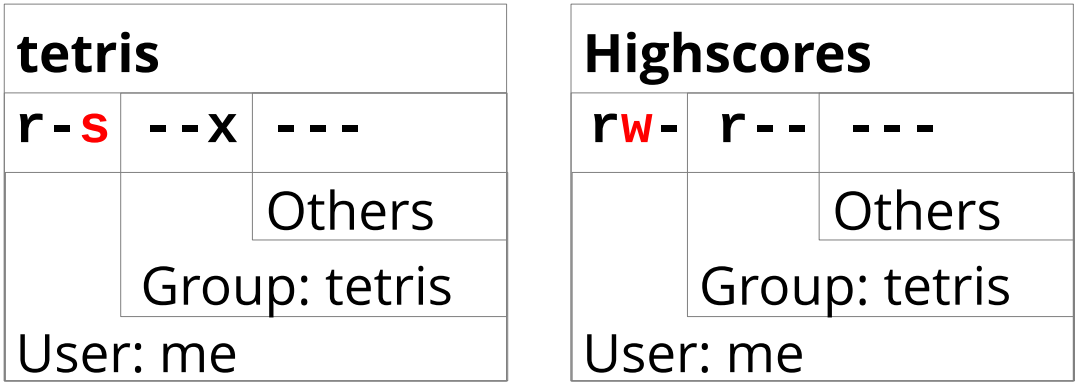
(Annahme: fritz ist zusätzlich in der Gruppe tetris)



Beispiel: Highscore-Liste



(Annahme: fritz ist zusätzlich in der Gruppe tetris)

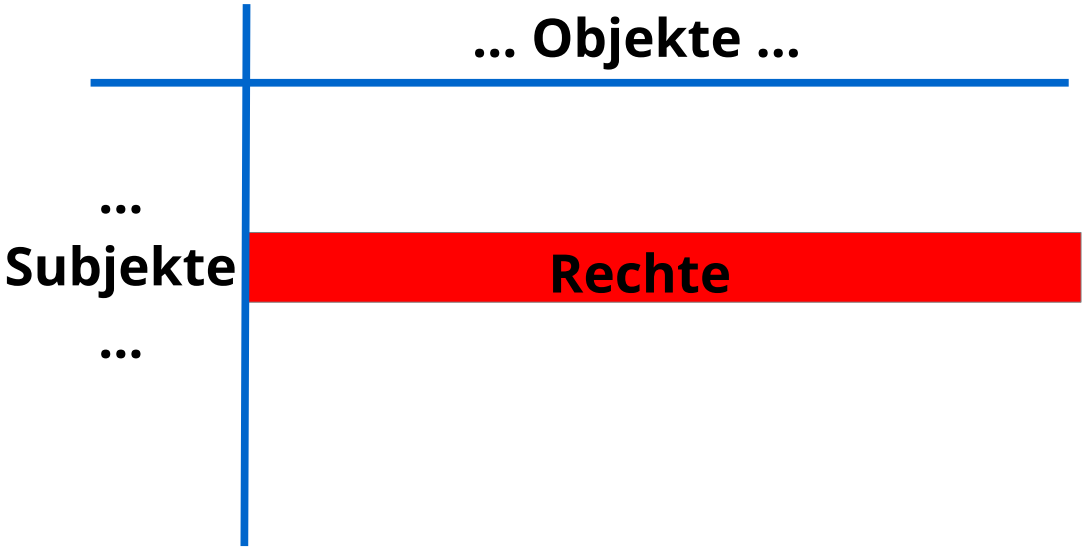


setuid-Probleme

- Erweiterung der Rechte eines Benutzers genau für den Fall der Benutzung dieses Programms
- „Besitzer“ des Programms vertraut dem Benutzer, wenn er dieses Programm nutzt
  - Besitzer kann Administrator, aber auch normaler Benutzer sein
- Problem: **Programmfehler**
  - können zu sehr großen Rechteerweiterungen führen
  - z.B. Shell-Aufruf aus einem solchen Programm heraus
- Praktische Erfahrung: immer noch zu viele Rechte!

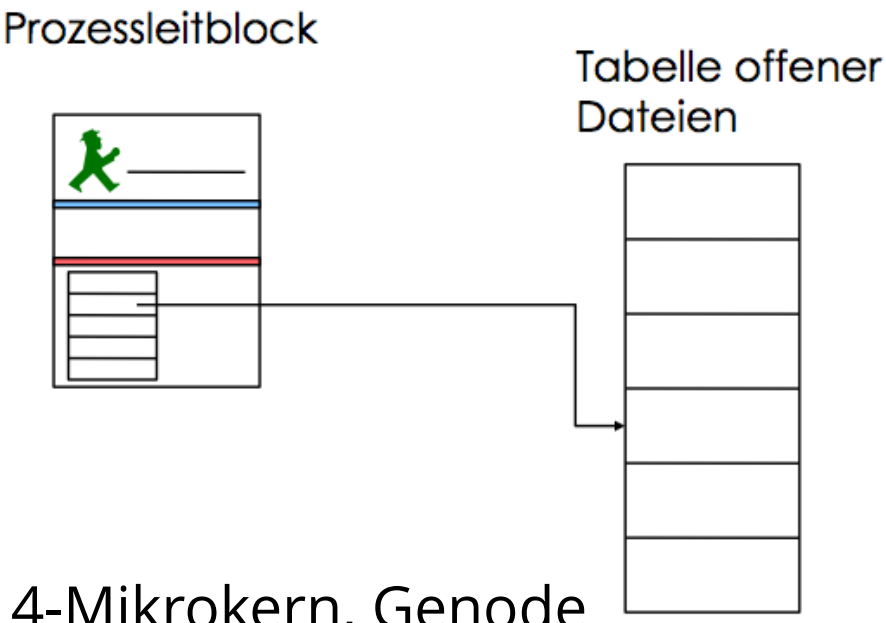
Capabilities

- Zeilenweise Darstellung der Schutzmatrix: *Capability*
  - pro Subjekt (bei Bedarf) **Liste von Objekten mit zugehörigen Rechten**
- Analogie: (nicht personalisierte) Eintrittskarte



Beispiel

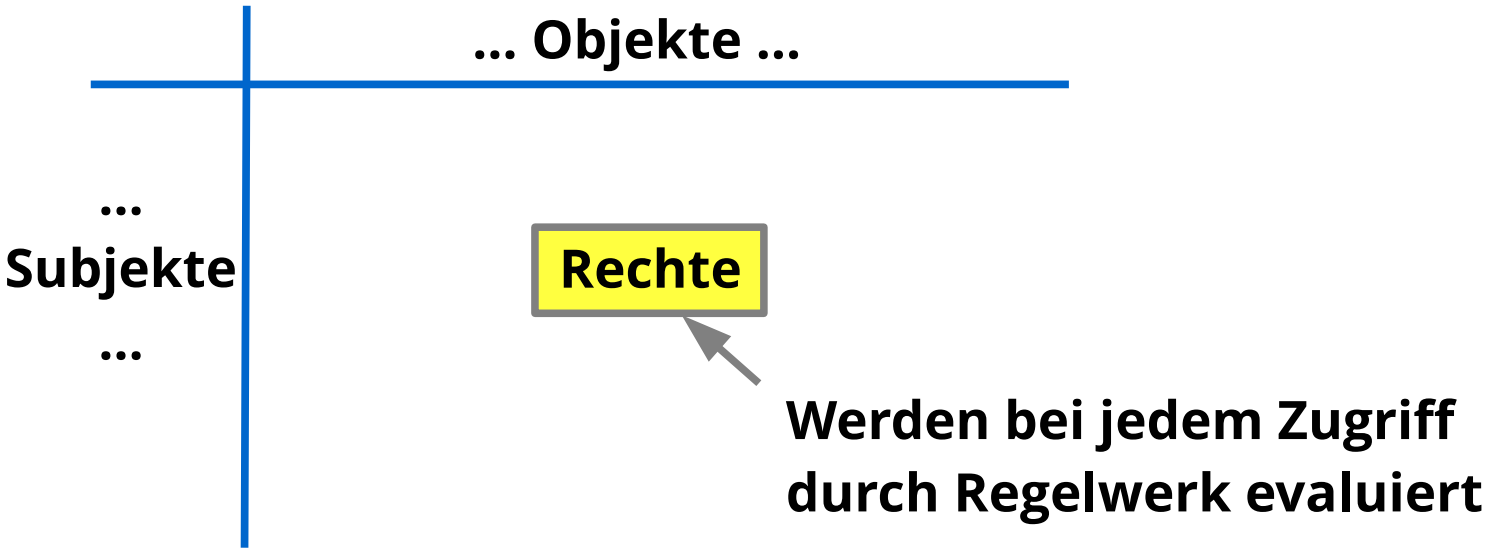
- Rudimentäre Form: Unix-Dateideskriptoren
- Weitergabe durch **fork**-Systemaufruf (oder Sockets)
  - Ermöglicht Zugriff auf Dateien ohne erneute Prüfung der UNIX-Zugriffsrechte
  - z.B. Foto-Viewer und separater „Powerbox“-Prozess
- kurzfristige Rechtevergabe also quasi mit *Capabilities*
- langfristig wie gehabt über ACLs



- Betriebssysteme mit vollwertigen *Capability*-Systemen: z.B. KeyKOS, L4-Mikrokern, Genode

Schutzmatrix regelbasiert

- ***MAC: Mandatory Access Control*** (regelbasierte Zugriffssteuerung)
- Konzept:
  - Subjekte und Objekte haben Attribute („*labels*“)
  - Entscheidung über Zugriff anhand von Regeln
- Implementierung in sog. Sicherheitskernen, z.B. SELinux



Zusammenfassung Rechteverwaltung

- existierende Umsetzungen der Zugriffsmatrix
  - spaltenweise: ACLs
  - zeilenweise: *Capabilities*
  - regelbasiert: *Mandatory Access Control*
- in gängigen Betriebssystemen Bestandteile **aller drei Varianten**

Inhalt

- Überblick über Sicherheitsprobleme
- Rechteverwaltung
  - Schutzmatrix
  - *Access Control Lists*
  - *Capabilities*
  - *Mandatory Access Control*
- **Systemsoftware und Sicherheit**
  - Hardwarebasierter Schutz
  - Softwarebasierter Schutz
- Zusammenfassung

Systemsoftware und Sicherheit

- Schutz auf Hardware-Seite
  - MMU
  - Schutzringe

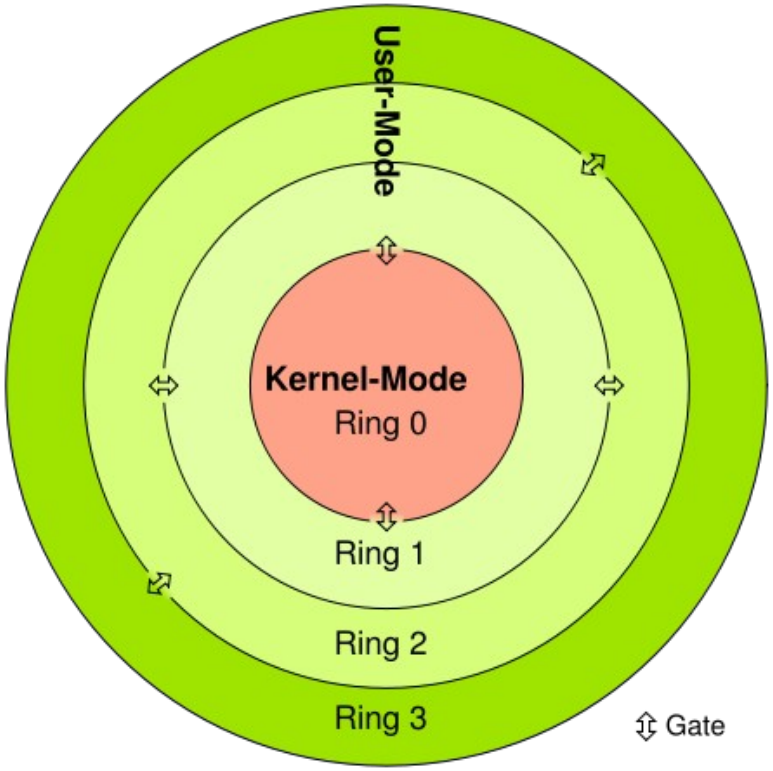
- … ergänzt durch Schutz auf Systemsoftware-Seite
  - Alleinige Kontrolle der Hardware durch das Betriebssystem
  - Alleinige Kontrolle über alle Prozesse
  - Alleinige Kontrolle über alle Ressourcen
  - Bereitstellung von
    - Identifikationsmechanismen
    - Authentisierungsmechanismen
    - Privilegseparation
    - Kryptographische Sicherung von Informationen

Hardwarebasierter Schutz: MMU

- ***Memory Management Unit***
  - Hardwarekomponente der CPU, die Zugriff auf Speicherbereiche umsetzt und kontrolliert
  - Umsetzung von Prozess-Sicht (virtuelle Adressen) auf Hardware-Sicht (physische Adressen)
- Einteilung des Hauptspeichers in „Seiten“ (*pages*)
- Schutz durch …
  - Einblendung nur der genau benötigten Menge an Speicherseiten des Hauptspeichers in den virtuellen Adressraum eines Prozesses
  - **Isolation** der physischen Adressräume unterschiedlicher Prozesse
  - Schutzbits für jede Seite, die bei jedem Zugriff kontrolliert werden
    - Lesen/Schreiben/Code ausführen
    - Zugriff im *User-Mode/Supervisor-Mode*

Schutzringe

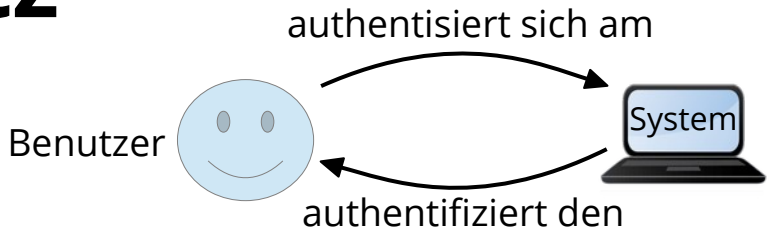
- Privilegienkonzept
  - Ausführung von Code ist bestimmtem Schutzring zugeordnet
  - Code in Ring 0 hat Zugriff auf alle Ressourcen des Systems
  - User-Programme laufen in Ring 3
  - Ringe 1 u. 2 für BS-nahen Code
    - z.B. Gerätetreiber
- Ringe schränken ein …
  - den nutzbaren **Befehlssatz** der CPU
    - z.B. keine Interruptsperren in Ring > 0
  - den zugreifbaren **Adressbereich** für den Prozess
    - Sperre von I/O-Zugriffen



Softwarebasierter Schutz

- Identifikationsmechanismen
- Unix: Benutzeridentifikation, Gruppenidentifikation
  - Numerischer Wert
  - Umgesetzt in Texte (Usernamen, Gruppennamen) durch *lookup* in `/etc/passwd`
- Ressourcen haben zugeordnete Besitzer
- Superuser: uid = 0
  - … hat alle Rechte im System.

Softwarebasierter Schutz



- Authentifizierungsmechanismen
  - Unix login
  - Abfrage von Benutzernamen und Passwort
  - Verifikation des Passworts mit im System hinterlegtem Passwort
    - Entweder durch Verschlüsselung des eingegebenen Passworts und Vergleich mit dem hinterlegten verschlüsselten Wertes
    - Oder durch Verifikation eines *Hash*-Wertes
  - Der login-Prozess startet dann das erste Benutzerprogramm (z.B. eine *shell*) mit der uid und gid, die zum eingegebenen Benutzernamen gehören

Softwarebasierter Schutz

- Kryptographische Sicherung von Informationen
  - z.B. Passwörter der Systembenutzer DES-verschlüsselt
  - Ursprünglich in Unix: `/etc/passwd`

```
root:4t6f4rt3423:0:0:System Administrator:/var/root:/bin/sh
daemon:ge53r3rfrg:1:1:System Services:/var/root:/usr/bin/false
me:1x3Fe5$gRd:1000:1000:Michael Engel:/home/me:/bin/bash
```

  - *Problem*: verschlüsselte Passwörter für alle Benutzer lesbar
    - … und mit genügend Zeit auch durch **„brute force“-Angriff** zu knacken
    - Fertige Tools wie z.B. *John the Ripper*

- Heute: Nur Benutzerinformationen in `/etc/passwd`
  - Passwörter stehen separat in `/etc/shadow`!

```
-rw-r--r-- 1 root root   1353 May 28 22:43 /etc/passwd
-rw-r----- 1 root shadow  901 May 28 22:43 /etc/shadow
```

Inhalt

- Überblick über Sicherheitsprobleme
- Rechteverwaltung
  - Schutzmatrix
  - *Access Control Lists*
  - *Capabilities*
  - *Mandatory Access Control*
- Systemsoftware und Sicherheit
  - Hardwarebasierter Schutz
  - Softwarebasierter Schutz
- **Zusammenfassung**

Fazit

- Sicherheit in vernetzten Umgebungen immer relevanter
  - Extrem hoher Schaden durch **Viren**, ***Phishing***, ***Botnets***, ***Ransomware***, …
- Umsetzung von zentralen **Entwurfsprinzipien**
  - geringstmögliche Privilegisierung, sichere Standardeinstellungen, Separierung von Privilegien
- **Schutz** durch Hardware- und Softwaremechanismen
  - **Isolation** mit Hilfe der MMU, Supervisor-/Usermode
  - Authentifizierung, Repräsentation des Nutzers in Prozessen
  - **Zugriffsmatrix** – Operation(Subjekt, Objekt)
    - Umsetzung in Kombination von ACLs, Capabilities und MAC
- (relativ) neu: Nicht einmal mehr auf die Hardware ist Verlass …
  - Stichworte ***„Meltdown“***, ***„Spectre“*** und Co.