Fachhochschule
Dortmund
University of Applied Sciences and Arts

# Informationssicherheit – SoSe 2023
# Zugriffskontrolle

Prof. Dr. Holger Schmidt
holger.schmidt004[at]fh-dortmund.de

Fachhochschule Dortmund
Fachbereich Informatik
Professur für IT-Sicherheit, Informatik

# Themen & Lernziele

# Themen



- Klassische Zugriffskontrollmodelle: DAC, MAC, RBAC
- Berechtigungsrichtlinien: Zugriffskontrollmatrix, ACL, Capabilities
- Weitergehende Zugriffskontrollmodelle, insb. ABAC, ReBAC, Chinese-Wall, Bell-LaPadula

# Lernziele



Die Studierenden sind in der Lage,

- Zugriffskontrollmodelle zu differenzieren und zu erklären.
- Berechtigungsrichtlinien als Zugriffskontrollmatrix, ACL und Capabilities zu modellieren.
- Linux Dateisystemberechtigungen zu analysieren und zu erstellen.

# Zugriffskontrolle

# Zugriffskontrolle



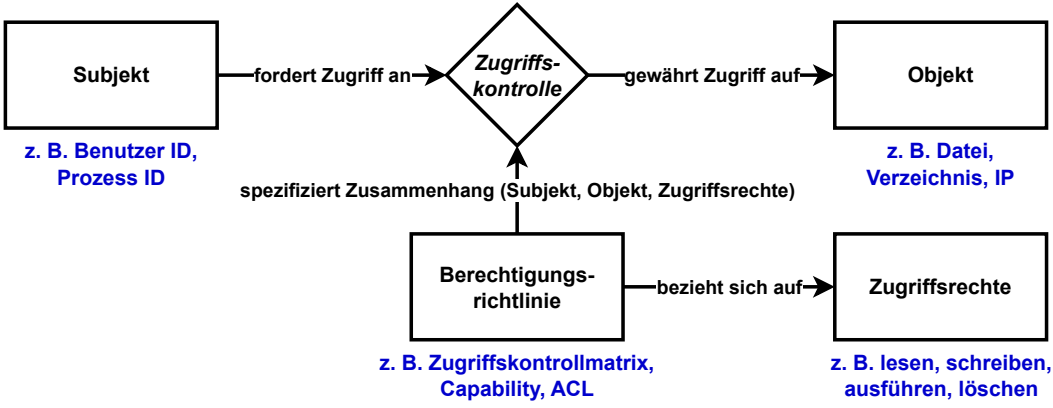


Abbildung selbst erstellt

# Zugriffskontrollmatrix-Modell



- **Zugriffskontrollmatrix** legt fest, welche **Operationen** (Zugriffsrechte) **Domänen** (Subjekte) auf **Objekten** ausführen dürfen.
- Beispiel: Linux nutzt **Domänen** `User`, `Group` und `Other` mit **Operationen** `Read`, `Write` und `Execute` auf `Datei` **Objekt**.

| object / domain | $F_1$ | $F_2$ | $F_3$ | printer |
|---|---|---|---|---|
| $D_1$ | read | | read | |
| $D_2$ | | | | print |
| $D_3$ | | read | execute | |
| $D_4$ | read write | | read write | |

Figure 17.5 aus Silberschatz et al., 2018

| Dir | User | Group | Other |
|---|---|---|---|
| d | r w x | r - x | - - x |
| | 4 2 1 | 4 1 | 1 |
| | 7 | 5 | 1 |

Linux Zugriffstabelle, entspricht Spalte in Zugriffskontrollmatrix; Tabelle selbst erstellt

# Access Control Lists (ACL)



| | bill.doc | edit.exe | fun.com |
|---|---|---|---|
| Alice | | {x} | {r, x} |
| Bob | {r, w} | {x} | {r, w, x} |

Tabelle selbst erstellt

- Objekt im Fokus
- **Spaltenweise** Betrachtung von Zugriffskontrollmatrix
- **ACL** ist Menge der Zugriffsrechte aller Subjekte für **ein Objekt**
- z. B. bill.doc: {Bob: r, w} und exit.exe: {Alice: x}, {Bob: x} und fun.com: {Alice: r, x}, {Bob: r, w, x}
- Vorteil: Ressourcenschonend (Speicher) bei umfangreicher Zugriffskontrollmatrix
- Nachteil: unübersichtlich

# Capabilities



| | bill.doc | edit.exe | fun.com |
|---|---|---|---|
| Alice | | {x} | {r, x} |
| Bob | {r, w} | {x} | {r, w, x} |

Tabelle selbst erstellt

- Subjekt im Fokus
- **Zeilenweise** Betrachtung von Zugriffskontrollmatrix
- **Capability** ist Menge der Zugriffsrechte **eines Subjektes** für alle Objekte
- z. B. Alice: {edit.exe: x}, {fun.com: r, x} und Bob: {bill.doc: r, w}, {edit.exe: x}, {fun.com: r, w, x}
- Vorteile: Delegation von Zugriffsrechten durch Kopieren von Capabilities, Löschen von Subjekten durch Löschen von Capabilities
- Nachteile: unübersichtlich, Ändern von Zugriffrechten für mehrere Objekte aufwändig

# Least Privilege & Deny by Default



Entwurfsprinzipien nach Smith (2012)

## Least privilege

- Subjekten wird eine minimale Menge von Zugriffsrechten für Objekte gewährt, die für ihre Aufgabenerfüllung unbedingt erforderlich sind.
- Beispiel: als Benutzer können nur Programme beendet werden, die zuvor von diesem Benutzer gestartet wurden

## Deny by default

- Gewährung keiner Zugriffrechte, außer denen, die durch Berechtigungsrichtlinie gefordert sind
- Beispiel: Voreinstellung bei Firewalls (jeglicher Netzwerkverkehr verboten, bis auf zu spezifizierende Ausnahmen)

# Firewall



- **Firewall** trennt vertrauenswürdige Domäne von nicht vertrauenswürdiger Domäne durch Filterung bzw. Blockierung von Netzwerkverkehr zwischen diesen Domänen
- **Netfilter**[1] ist Linux Firewall/Paketfilter mit **nftables**[2]
  - **Regelwerk** besteht aus **Tabellen** (`table`), **Chains** (`chain`), **Regeln**
  - Tabellen kategorisieren Regeln, z. B. `filter` für Paketfilterung
  - Chains sind Container für Regeln und steuern Reihenfolge der Anwendung von Regeln
  - `nft list ruleset` gibt das aktive Regelwerk aus.
  - `nft flush ruleset` löscht alle Tabellen, Chains und Regeln.

[1] https://www.netfilter.org/, abgerufen am 28. Juni 2023
[2] https://www.netfilter.org/projects/nftables/index.html, abgerufen am 28. Juni 2023

# Netfilter/nftables Beispiel



```
#!/usr/sbin/nft -f
flush ruleset

# Tabelle mit Namen filter von Typ inet
table inet filter {
 # Kette mit Namen input
 chain input {
  # type filter Netzwerkverkehr wird gefiltert
  # hook input eingehender Netzwerkverkehr wird gefiltert
  # policy drop setzt deny by default um
  type filter hook input priority 0; policy drop;
  # TCP-Verbindungen auf Standard-Port FTP sind erlaubt
  tcp dport ftp accept
  }
}
```

# Discretionary Access Control (DAC)



- Bei **DAC** basiert die Zugriffskontrolle darauf, welches Subjekt in **Besitz** eines Objektes ist. Subjekte bestimmen die Zugriffsrechte *ihrer* Objekte selbst.
- Vorteile: einfach, Delegation
- Nachteil: z. B. Besitzer einer Datei kann Zugriffsrechte selbst verändern, was häufig zu Verwundbarkeiten führt

# Mandatory Access Control (MAC)



- **MAC** verhält sich gegensätzlich zu DAC: Subjekte haben keinen Einfluss auf Zugriffsrechte
- Zentrale Instanz realisiert Zugriffskontrolle basierend auf Berechtigungsrichtlinie.
- Vorteile: Nachteil von DAC ausgeräumt
- Nachteil: kompliziert (Policy-Spezifikation)

# Role-Based Access Control (RBAC)



- **RBAC** (Ferraiolo & Kuhn, 1992) bindet **Privilegien** (Zugriffsrechte) an **Rollen**.
- Ein Subjekt kann eine oder mehrere Rollen besitzen.
- Vorteil: Rechteverwaltung einfach

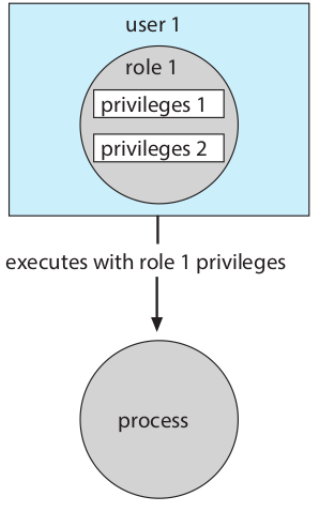


Figure 17.10 aus Silberschatz et al., 2018

# SELinux Policy Beispiel



- **Security-Enhanced Linux (SELinux)**[3] ist eine Implementierung von MAC im Linux-Kernel mithilfe von Policies.
- In Kombination mit RBAC

```
policy_module(admin_logs_policy, 1.0)

gen_require(‘
    type sysadm_r;
’)

# Allow system admin to list the contents of /var/log
logging_list_logs(sysadm_r)
```

[3] https://github.com/SELinuxProject, aufgerufen am 28. Juni 2023

# Linux Dateisystemberechtigungen



- Es gibt **drei Gruppen** (wie Rollen):
  - **User, Group, Other**
- Sowie **drei Zugriffsrechte**:
  - **(R)ead, (W)rite, E(x)ecute**
- Datei wird Gruppe sowie Besitzer nach dem Schema `group:user` zugeordnet

| Dir | User | Group | Other |
|---|---|---|---|
| d | r w x | r - x | - - x |
| | 4 2 1 | 4 1 | 1 |
| | 7 | 5 | 1 |

Tabelle selbst erstellt

```
>ls -l
-rwxr-x--x 1 user group 0
01. Jan 00:00 script.sh
```

## Änderung von Zugriffsrechten

Mit `chmod` können die Zugriffsrechte angepasst werden. Mit z. B. `chmod 751 script.sh` hat nur der Besitzer alle Berechtigungen, Gruppenmitglieder dürfen die Datei nicht verändern und alle anderen nur ausführen.

# Attribute-Based Access Control (ABAC)



- **ABAC** (NIST SP 800-162, 2019) bindet Zugriffsrechte an **Attribute** von Subjekten und Objekten, z. B. Rolle, Projekt, Tageszeit, MAC Adresse.
- D. h. Zugriffsrecht wird durch Attribute mit einem **Kontext** versehen
- Mittlerweile verbreitet, z. B. in Frameworks wie Spring Security[4]
- Vorteile: Role explosion Problem von RBAC ausgeräumt, meist einfacher zu verwalten als RBAC, Zugriffsrechte fein justierbar (Least Privilege)
- Im Allgemeinen sollte (falls anwendbar) ABAC der Vorzug vor RBAC gegeben werden.

[4] https://dzone.com/articles/simple-attribute-based-access-control-with-spring, aufgerufen am 28. Juni 2023

# Weitere Zugriffskontrollmodelle



**Relationship-Based Access Control (ReBAC)** von Gates (2007) basiert auf Beziehungen (über Subjekte und Objekte), angewandt in sozialen Netzwerken

**Chinese-Wall Modell** von Brewer und Nash (1989) realisiert kommerzielle Strategie, angewandt im Finanzsektor

**Bell-LaPadula** von Bell und LaPadula (1973) Hierarchie-orientiert, angewandt im militärischen Bereich

# Zusammenfassung

# Zusammenfassung



- Grundlagen Zugriffskontrolle erklärt
- Spezifikation Berechtigungsrichtlinien gelernt
- Klassische und weitergehende Zugriffskontrollmodelle vorgestellt

# Weiterführende Literatur

# Weiterführende Literatur



- *IT-Sicherheit – Konzepte - Verfahren - Protokolle*, Kapitel 6 von Eckert (2023)
- *OWASP Authorization Cheat Sheet*[5]
- *Security Engineering: A Guide to Building Dependable Distributed Systems*, Kapitel 6 von Anderson (2020)
- *A Survey Of Access Control Models*[6]

[5] https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html, aufgerufen am 28. Juni 2023

[6] https://csrc.nist.gov/csrc/media/events/privilege-management-workshop/documents/pvm-model-survey-aug26-2009.pdf, aufgerufen am 28. Juni 2023

# Literatur I




- Anderson, R. (2020). *Security Engineering: A Guide to Building Dependable Distributed Systems* (3. Aufl.). John Wiley & Sons Inc. https://www.cl.cam.ac.uk/~rja14/book.html (siehe S. 23).
- Bell, D. E., & LaPadula, L. J. (1973). *Secure Computer Systems: Mathematical Foundations and Model*. Mitre Corporation. (Siehe S. 19).
- Brewer, D. F. C., & Nash, M. J. (1989).The Chinese Wall Security Policy. *IEEE Symposium on Security and Privacy*, 206–214 (siehe S. 19).
- Eckert, C. (2023). *IT-Sicherheit: Konzepte - Verfahren - Protokolle* (11. Aufl.). De Gruyter Oldenbourg. (Siehe S. 23).
- Ferraiolo, D. F., & Kuhn, D. R. (1992).Role-Based Access Controls. *15th National Computer Security Conference*, 554–563. https://csrc.nist.gov/CSRC/media/Publications/conference-paper/1992/10/13/role-based-access-controls/documents/ferraiolo-kuhn-92.pdf (siehe S. 15).

# Literatur II




- Gates, C. E. (2007).Access Control Requirements for Web 2.0 Security and Privacy. *IEEE Security and Privacy Workshop on Web 2.0 Security and Privacy* (siehe S. 19).
- NIST SP 800-162. (2019). NIST Special Publication 800-162. Guide to ABAC Definition and Considerations. https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-162.pdf (siehe S. 18).
- Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts* (10th). John Wiley & Sons, Inc. (Siehe S. 7, 15).
- Smith, R. E. (2012).A Contemporary Look at Saltzer and Schroeder's 1975 Design Principles. *IEEE Security & Privacy*, *10*(6), 20–25 (siehe S. 10).