

2 Beispiel: UNIX

- Zugriffslisten für
 - ◆ Dateien
 - ◆ Shared memory-Segmente
 - ◆ Message queues
 - ◆ Semaphore
 - ◆ etc.
- Berechtigungen:
 - ◆ Lesen (*read*), Schreiben (*write*), Ausführen (*execute*)
 - ◆ für Besitzer, Gruppe und alle anderen unterscheidbar
- Subjekte:
 - ◆ Prozesse
 - ◆ Besitzer (Benutzer) und Zugehörigkeit zu einer oder mehreren Gruppen

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-Security.doc: 1998-01-28 10:40

L.16

Reproduktion oder die auch Verwendung dieser Vorlesung ist in der Universität Erlangen-Nürnberg gestattet. Bei der Zustimmung des Autors.

2 Beispiel: UNIX

- *Superuser*
 - ◆ Benutzer *root* hat automatisch alle Zugriffsrechte
- S-Bit-Programme
 - ◆ S-Bit ist ein besonderes Recht auf der Binärdatei des Programms
 - ◆ Besitzer der Datei wird bei der Ausführung auch Besitzer des Prozesses (sonst wird Aufrufer Besitzer des Prozesses)
- ★ Vorteil
 - ◆ Bereitstellen von Prozessen, die kontrolliert Aufrufen höhere Zugriffsberechtigungen erlauben
- ▲ Nachteil
 - ◆ Fehler im Prozess gibt Aufrufer volle Rechte des Programmbesitzers
 - ◆ fatal, falls das Programm *root* gehört

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-Security.doc: 1998-01-28 10:40

L.17

Reproduktion oder die auch Verwendung dieser Vorlesung ist in der Universität Erlangen-Nürnberg gestattet. Bei der Zustimmung des Autors.

3 Implementierung

- Globale Tabelle/Matrix
 - ◆ System hält eine Datenstruktur und prüft im betreffenden Eintrag die Berechtigungen
 - ◆ Tabelle üblicherweise recht groß: paßt evtl. nicht in den Speicher
- Zugriffslisten an den Objekten
 - ◆ jedes Objekt hält eine Liste der Berechtigungen (z.B. Unix Datei: Inode)
 - ◆ verringert üblicherweise den Platzbedarf für die Einträge (unnötige Felder der Matrix werden nicht repräsentiert)
- Zugriffslisten an den Subjekte
 - ◆ jedes Subjekt hält eine Liste von Objekten und den Berechtigungen, die das Subjekt für das Objekt hat
 - ◆ ähnlich Capabilities

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-Security.doc: 1998-01-28 10:40

L.18

Reproduktion oder die auch Verwendung dieser Vorlesung ist in der Universität Erlangen-Nürnberg gestattet. Bei der Zustimmung des Autors.

I.3 Schutzmodell nach Bell-La Padula

- Sicherheitsgrad
 - ◆ Tupel: ([Geheimhaltungsstufe](#), [Schutzkategorie](#))
 - ◆ Geheimhaltungsstufe: ein Element aus einer vollständig geordneten Menge (z.B. vertraulich, geheim, streng geheim)
 - ◆ Schutzkategorie: Teilmenge von systemspezifischen Sachgebieten (z.B. Arbeiter, Angestellte, leit. Angestellte, Post)
- ◆ Jedem Objekt und jedem Subjekt ist ein Sicherheitsgrad zugeordnet
- Sicherheitseigenschaft
 - ◆ Ein Subjekt kann nur Objekte mit gleichem oder niedrigerem Sicherheitsgrad lesen oder schreibend zugreifen.
 - ◆ Dabei gilt: $(g,s) \leq (g',s') \Rightarrow g \leq g' \wedge s \subseteq s'$

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-Security.doc: 1998-01-28 10:40

L.19

Reproduktion oder die auch Verwendung dieser Vorlesung ist in der Universität Erlangen-Nürnberg gestattet. Bei der Zustimmung des Autors.

I.3 Schutzmodell nach Bell-La Padula

- *-Eigenschaft
- ◆ Ein Subjekt kann nur dann gleichzeitig zu einem Objekt A lesen und zu einem Objekt B schreibenden Zugriff haben, wenn B den gleichen oder einen höheren Sicherheitsgrad besitzt als A
- ★ Es ist nicht möglich Informationen eines hohen Sicherheitsgrads zu einem Objekt niedrigeren Sicherheitsgrads zu transportieren

1 Beispiel

- Subjekte und Objekte mit Sicherheitsgraden
- ◆ D_{LA} = Personaldaten der leitenden Angestellten:
(streng geheim, {})
- ◆ D_{AN} = Personaldaten der sonstigen Angestellten:
(geheim, {})
- ◆ D_{AR} = Personaldaten der Arbeiter:
(geheim, {})
- ◆ S_{pers} = Leiter des Personalbüros:
(streng geheim, {Post, leit. Angestellte, Arbeiter, Angestellte})
- ◆ S_{stellv} = Sachbearb. leitende Angestellte, stellvertr. Leiter Personalbüro:
(streng geheim, {leit. Angestellte})
- ◆ S_{sach} = Sachbearbeiter Angestellte u. Arbeiter:
(geheim, {Arbeiter, Angestellte})
- ◆ S_{post} : Poststelle:
(streng geheim, {Post})

1 Beispiel (2)

- Prozeduren mit Sicherheitsgraden
- ◆ R_{LA} = Lesen von Pers.-Nr. und Lohn-/Gehaltsgr. aus D_{LA} :
(streng geheim, {leit. Angestellte})
- ◆ $R_{AN/AR}$ = Lesen von Pers.-Nr. und Lohn-/Gehaltsgr. aus D_{AN} oder D_{AR} :
(geheim, {Angestellte, Arbeiter})
- ◆ R_{post} = Lesen von Name, Abteilung und Pers.-Nr.:
(streng geheim, {Post})

1 Beispiel (3)

- Informationsflußkontrollgraph:
- R_{post}
(streng geheim, {Post})

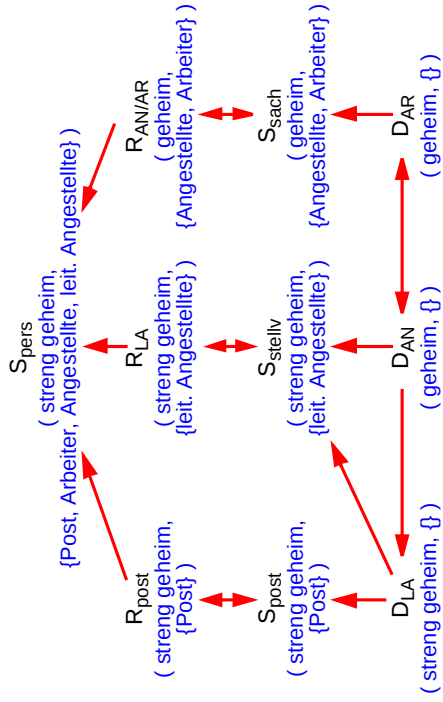
R_{LA}
(streng geheim, {leit. Angestellte})

$R_{AN/AR}$
(geheim, {Angestellte, Arbeiter})
- S_{post}
(streng geheim, {Post})
- S_{stellv}
(streng geheim, {leit. Angestellte})
- S_{sach}
(geheim, {Angestellte, Arbeiter})

 D_{LA}
(streng geheim, {}) D_{AN}
(geheim, {}) D_{AR}
(geheim, {})

1 Beispiel (4)

■ Informationsflußkontrollgraph:



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-Security.doc: 1998-01-28 10:40

I.24

Reproduction of this document for other purposes without the written permission of the author is prohibited.

1 Beispiel (4)

■ Sicherheitsgrade verhindern bestimmte Informationsflüsse unabhängig von der Schutzmatrix

◆ z.B. kann Information aus R_{post} nach S_{post} gelangen, von dort aber nicht mehr an S_{sach} weitergegeben werden

■ Umgekehrt kann die Schutzmatrix Einschränkungen treffen, die nicht durch die Sicherheitsgrade allein verhindert werden

◆ z.B. kann S_{stelv} nicht den kompletten Inhalt von D_{LA} lesen, obwohl der Informationsflußkontrollgraph dies erlauben würde

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-Security.doc: 1998-01-28 10:40

I.25

Reproduction of this document for other purposes without the written permission of the author is prohibited.

2 Bewertung

▲ Probleme

◆ Information erlangt immer höhere Sicherheitsgrade und kann dann nicht mehr weitergegeben werden

◆ Beispiel: Programm zur Steuererklärung greift auf streng geheime Buchhaltungsdaten zu → Steuererklärung ist streng geheim

■ Einführung von vertrauenswürdigen Prozeduren, die die *-Eigenschaft umgehen können

◆ Informationen können im Sicherheitsgrad wieder heruntergestuft werden

▲ vertrauenswürdige Prozeduren stellen wiederum ein Sicherheitsrisiko dar

◆ Verifikation nötig, aber schwierig

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-Security.doc: 1998-01-28 10:40

I.26

Reproduction of this document for other purposes without the written permission of the author is prohibited.

I.4 Schutz durch Speicherverwaltung

■ Schutz vor gegenseitigem Speicherzugriff

◆ Segmentierung und Seitenadressierung erlauben es, jedem Prozeß nur den benötigten Speicher einzublenden

◆ Segmentverletzung löst Unterbrechung aus

■ Systemaufrufe

◆ definierter Weg von einer Schutzumgebung (der des Prozesses) in eine andere (der des Betriebssystemes)

■ Erweiterung dieses Konzepts:

◆ allgemeine Prozeduraufrufe zwischen verschiedenen Schutzumgebungen, realisiert mit der Speicherverwaltung und deren Hardware (MMU)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998

I-Security.doc: 1998-01-28 10:40

I.27

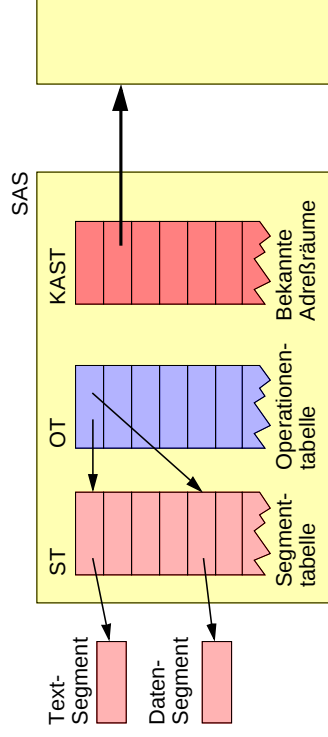
Reproduction of this document for other purposes without the written permission of the author is prohibited.

Modulkonzept von Habermann

- Idee (von 1976)
- Adreßräume (Module) bilden Schutzumgebungen
- Adreßräume bieten definierte Operationen an (ähnlich wie das Betriebssystem Systemaufrufe anbietet)
- Parameter werden in speziellen Segmenten übergeben
- Bietet allgemeinen Schutz der Module und erlaubt kontrollierte Interaktionen
- Module besitzen einen statischen Adreßraum (SAS, *Static address space*)
- enthält Liste von Segmenten, die zu dem Modul gehören bzw. von dem Modul zugegriffen werden dürfen
- enthält Liste von angebotenen Operationen mit den Angaben, welche Segmente jede Operation benötigt (u.a. Segment für die auszuführenden Instruktionen)

Modulkonzept von Habermann (2)

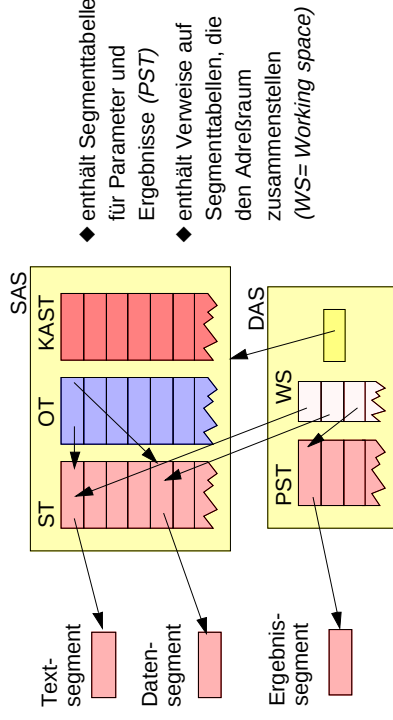
- enthält Liste von bekannten Adreßräumen anderer Module (dort können dann Operationen aufgerufen werden)



KAST = *Known address space table*

Modulkonzept nach Habermann (3)

- Aktivitätsträger sind einen dynamische Adreßraum zugeordnet (DAS, *Dynamic address space*)



- enthält Segmenttabelle für Parameter und Ergebnisse (PST)
- enthält Verweise auf Segmenttabellen, die den Adreßraum zusammenstellen (WS= *Working space*)

Beispielaufwurf

- SAS des Benutzers ruft Operation „open“ des SAS des Dateisystems auf

