

Zugriffskontrolle

Sicherheit und technischer Datenschutz in Informationssystemen
Seminar im Sommersemester 2006

Dominik Leonhardt

dominik@dleo.de

Betreuung: Jutta Mülle

IPD – Lehrstuhl Prof. Böhm, Universität Karlsruhe

26.06.2006

Wofür benötigt man Zugriffskontrolle?



Anne



Anne speichert alle ihre persönlichen Dokumente auf ihrem privaten PC.

Wofür benötigt man Zugriffskontrolle?



Anne



Niemand außer Anne hat Zugriff auf den PC.

Wofür benötigt man Zugriffskontrolle?

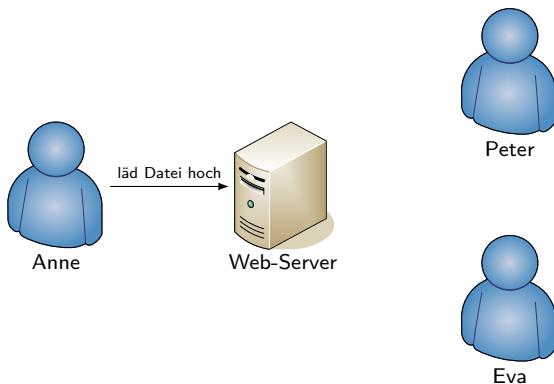


Anne



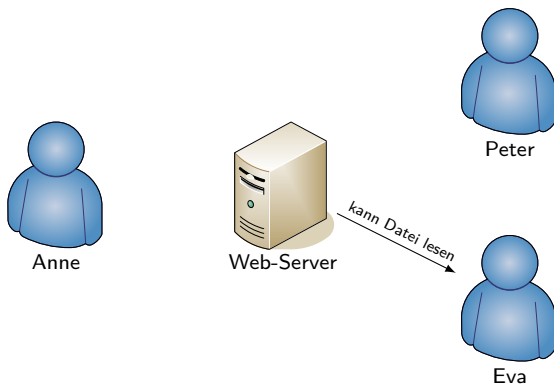
Wieso benötigt Anne eine Zugriffskontrolle?

Wofür benötigt man Zugriffskontrolle?



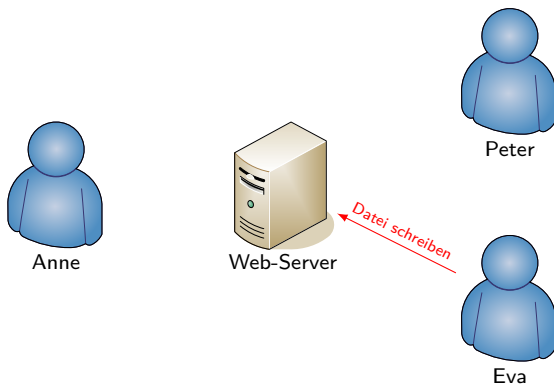
Anne hat Speicherplatz auf einem Web-Server und hinterlegt dort eine Datei für Eva.

Wofür benötigt man Zugriffskontrolle?



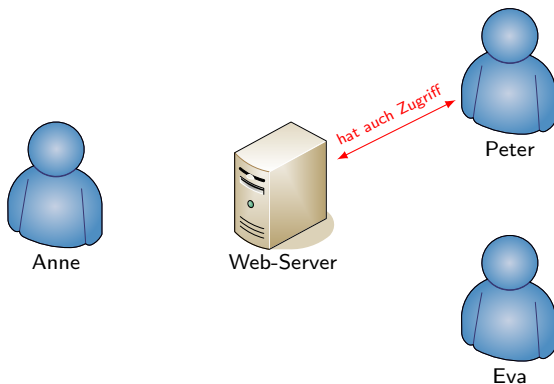
Eva kann die Datei auf ihren PC herunterladen und lesen.

Wofür benötigt man Zugriffskontrolle?



Eva kann aber die Datei auch schreiben,
was Anne eigentlich nicht will.

Wofür benötigt man Zugriffskontrolle?



Peter soll die Datei eigentlich weder lesen noch schreiben können.

Wofür benötigt man Zugriffskontrolle?



Anne braucht dringend eine gute Zugriffskontrolle.

Inhalt

① Das Modell für die Zugriffskontrolle

Zugriffskontrollmatrix

Ist ein System sicher?

Take-Grant Schutzmodell

Schematic Protection Modell

② Die Mechanismen

Zugriffskontrollliste (ACL)

ACL in Windows

ACL in Linux/POSIX

Befähigungen

Schlösser und Schlüssel

Ringbasierte Zugriffskontrolle

Propagierte Zugriffskontrolle

③ Zusammenfassung

Zugriffskontrollmatrix (Access Control Matrix oder ACM)

- Geschützte Elemente heißen *Objekte* (Ressourcen, ...)
- Aktive Objekte heißen *Subjekte* (Prozesse, Benutzer, ...)
- Eine Menge von *Rechten* (schreiben, lesen, ausführen, ...)

Die Verbindung zwischen *Objekten* und *Subjekten* werden in einer zweidimensionalen Matrix vermerkt.

	Datei 1	Datei 2	Anne	Peter
Anne	r,w,o	r	r,w,x,o	r
Peter	r	r,w,o	w	r,w,x,o

Tabelle: Beispiel einer ACM ¹

¹r = lesen, w = schreiben, x = ausführen, o = besitzen

Ist ein System sicher?

Problem

Die Grundlegende Frage: Ist ein gegebenes System sicher?

Was heißt sicher?

Definition

Wenn ein generisches Recht r zu einem Element in der Zugriffskontrollmatrix hinzugefügt wird, das es bisher nicht besessen hat, dann nennen wir dieses Recht *entwichen* (*leaked*).

Wir unterscheiden nicht zwischen *entwichen* und *authorisiertem Transfer* von Rechten.

Was heißt sicher?

Definition

Wenn ein generisches Recht r zu einem Element in der Zugriffskontrollmatrix hinzugefügt wird, das es bisher nicht besessen hat, dann nennen wir dieses Recht *entwichen* (*leaked*).

Wir unterscheiden nicht zwischen *entwichen* und *authorisiertem Transfer* von Rechten.

Was heißt sicher?

Definition

Wenn in einem System zu keiner Zeit das Recht r entweichen kann, dann nennen wir dieses System *sicher bezüglich des Rechtes r* (*safe with respect to the right r*).

Sicher ist aber nicht gleich sicher!

Die englische Literatur unterscheidet zwischen:

„safe“

- bezieht sich auf ein abstraktes Modell
- kann formal untersucht werden

„secure“

- bezieht sich auf die Implementierung
- ein „safe“ System kann „unsecure“ sein

Was heißt sicher?

Definition

Wenn in einem System zu keiner Zeit das Recht r entweichen kann, dann nennen wir dieses System *sicher bezüglich des Rechtes r* (*safe with respect to the right r*).

Sicher ist aber nicht gleich sicher!

Die englische Literatur unterscheidet zwischen:

„safe“

- bezieht sich auf ein abstraktes Modell
- kann formal untersucht werden

„secure“

- bezieht sich auf die Implementierung
- ein „safe“ System kann „unsecure“ sein

Was heißt sicher?

Definition

Wenn in einem System zu keiner Zeit das Recht r entweichen kann, dann nennen wir dieses System *sicher bezüglich des Rechtes r* (*safe with respect to the right r*).

Sicher ist aber nicht gleich sicher!

Die englische Literatur unterscheidet zwischen:

„safe“

- bezieht sich auf ein abstraktes Modell
- kann formal untersucht werden

„secure“

- bezieht sich auf die Implementierung
- ein „safe“ System kann „unsecure“ sein

Safety-Problem für die allgemeine ACM

Problem

Gibt es einen Algorithmus, der entscheiden kann ob ein gegebenes System sicher bezüglich eines Rechtes r ist?

Die schlechte Nachricht ist:

Theorem (Harrison, Ruzzo und Ullman, 1976)

Das Safety-Problem ist im Allgemeinen nicht entscheidbar.

Safety-Problem für die allgemeine ACM

Problem

Gibt es einen Algorithmus, der entscheiden kann ob ein gegebenes System sicher bezüglich eines Rechtes r ist?

Die schlechte Nachricht ist:

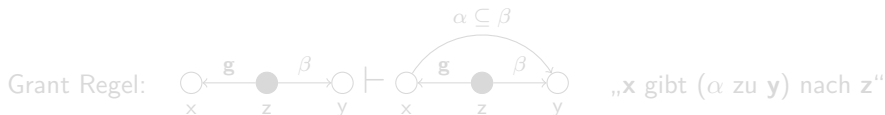
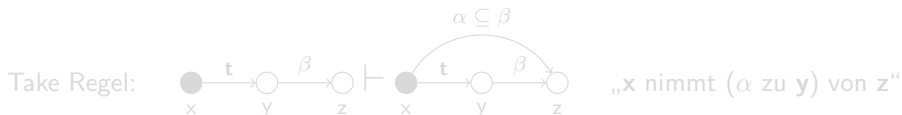
Theorem (Harrison, Ruzzo und Ullman, 1976)

Das Safety-Problem ist im Allgemeinen nicht entscheidbar.

Take-Grant Schutzmodell

Gibt es ein Sicherheitsmodell, für das das Safety-Problem entscheidbar ist?

Ja, das Take-Grant Schutzmodell:

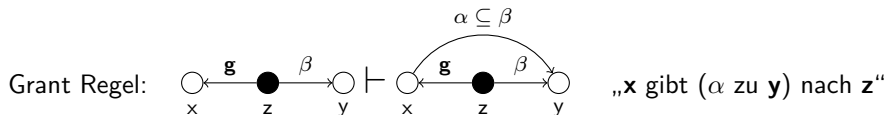
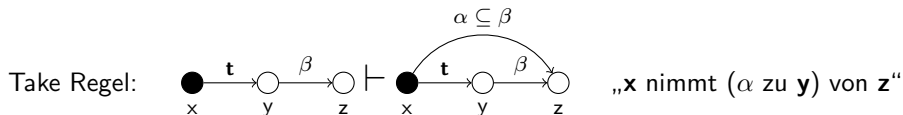


- Safety-Problem entscheidbar (in linearer Zeit zur Größe des Graphen)
- Nicht so mächtig wie ACM

Take-Grant Schutzmodell

Gibt es ein Sicherheitsmodell, für das das Safety-Problem entscheidbar ist?

Ja, das Take-Grant Schutzmodell:

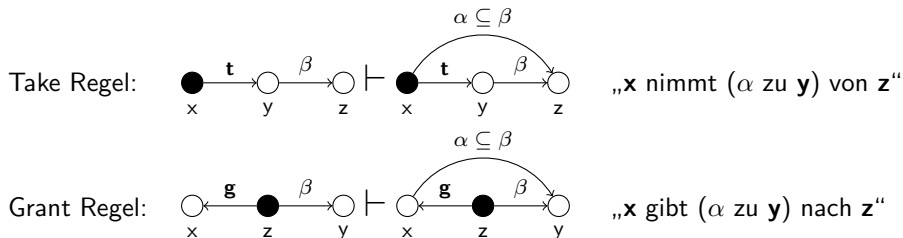


- Safety-Problem entscheidbar (in linearer Zeit zur Größe des Graphen)
- Nicht so mächtig wie ACM

Take-Grant Schutzmodell

Gibt es ein Sicherheitsmodell, für das das Safety-Problem entscheidbar ist?

Ja, das Take-Grant Schutzmodell:



- Safety-Problem entscheidbar (in linearer Zeit zur Größe des Graphen)
- Nicht so mächtig wie ACM

Schematic Protection Modell

- T ist die Menge der Typen, nach denen sich die Anwendung der Rechte unterscheiden (*protection types*)
- $T = TS \cup TO$, wobei TS die Subjekt-Typen und TO die Objekt-Typen sind
- \mathbf{X} ist eine eindeutige Einheit (Subjekt oder Objekt)
- \mathbf{X}/r ist ein Ticket das dem Besitzer das Recht r auf die Einheit \mathbf{X} gibt
- \mathbf{X}/rc ist das copy-Ticket für das Recht r auf die Einheit \mathbf{X}

Schematic Protection Modell

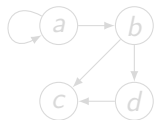
Die Menge *can-create* $cc \subseteq TS \times T$ legt fest welche Subjekte welche Einheiten erzeugen können.

Beispiel: *can-create* = $\{(users, files), (root, users)\}$

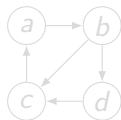
Definition

Ein Schema ist *azyklisch* (acyclic), wenn der *can-create* Graph keine Zyklen, außer von einem Knoten zu sich selbst, hat.

Beispiel:



azyklisch



nicht azyklisch

Schematic Protection Modell

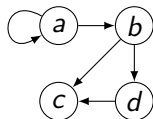
Die Menge *can-create* $cc \subseteq TS \times T$ legt fest welche Subjekte welche Einheiten erzeugen können.

Beispiel: *can-create* = $\{(users, files), (root, users)\}$

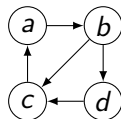
Definition

Ein Schema ist *azyklisch* (acyclic), wenn der *can-create* Graph keine Zyklen, außer von einem Knoten zu sich selbst, hat.

Beispiel:



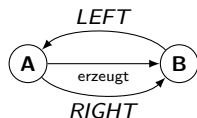
azyklisch



nicht azyklisch

Schematic Protection Modell

create-rule $cr(a, b)$ gibt an welche Tickets von B an A übergeben werden (*LEFT*). Ist B ein Subjekt, dann wird außerdem angegeben welche Tickets von A an B übergeben werden (*RIGHT*).



$cr(a, b) = LEFT | RIGHT$ mit
 $LEFT, RIGHT \subseteq \{a/x:c, self/x:c \mid x:c \in R\}$

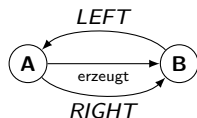
Definition

Ein Schema ist *abschwächend* (attenuating), wenn für alle $cr(a, a)$ mit $(a, a) \in cc$ gilt:

- 1 $RIGHT \subseteq LEFT$
- 2 Nur Tickets $A'/x:c$ oder $A/x:c$, für die das $A/x:c$ Ticket bereits beim Erzeuger vorhanden war, werden übergeben.

Schematic Protection Modell

create-rule $cr(a, b)$ gibt an welche Tickets von B an A übergeben werden (*LEFT*). Ist B ein Subjekt, dann wird außerdem angegeben welche Tickets von A an B übergeben werden (*RIGHT*).



$cr(a, b) = LEFT | RIGHT$ mit
 $LEFT, RIGHT \subseteq \{a/x:c, self/x:c \mid x:c \in R\}$

Definition

Ein Schema ist *abschwächend* (attenuating), wenn für alle $cr(a, a)$ mit $(a, a) \in cc$ gilt:

- 1 $RIGHT \subseteq LEFT$
- 2 Nur Tickets $A'/x:c$ oder $A/x:c$, für die das $A/x:c$ Ticket bereits beim Erzeuger vorhanden war, werden übergeben.

Schematic Protection Modell

- Für azyklische, abgeschwächte Schemata (acyclic attenuating schemes) ist das Safety-Problem in polynomialer Zeit entscheidbar
- SPM ist mächtiger als das Take-Grant Modell und mit kleinen Erweiterungen (ESPM) äquivalent zum ACM Modell

Inhalt

① Das Modell für die Zugriffskontrolle

Zugriffskontrollmatrix

Ist ein System sicher?

Take-Grant Schutzmodell

Schematic Protection Modell

② Die Mechanismen

Zugriffskontrollliste (ACL)

ACL in Windows

ACL in Linux/POSIX

Befähigungen

Schlösser und Schlüssel

Ringbasierte Zugriffskontrolle

Propagierte Zugriffskontrolle

③ Zusammenfassung

Zugriffskontrollliste (Access Control List oder ACL)

- Weitverbreitete Möglichkeit Zugriffskontrolle zu implementieren
- Basiert auf der Zugriffskontrollmatrix
- Jedes Objekt/Subjekt hat eine eigene Liste mit Zugriffsrechten:

```
acl(Datei 1)= { (Prozess 1, {read, write, own}), (Prozess 2, {append})}  
acl(Datei 2)= { (Prozess 2, {read, write, own} )}
```

- Wenig Speicherbedarf
- Schneller Zugriff
- Schnelles Löschen und Hinzufügen von Objekten

Bei vielen Benutzern kann die Liste lang werden, deshalb:

- Wildcards (*)
- Benutzergruppen

Zugriffskontrollliste (Access Control List oder ACL)

- Weitverbreitete Möglichkeit Zugriffskontrolle zu implementieren
- Basiert auf der Zugriffskontrollmatrix
- Jedes Objekt/Subjekt hat eine eigene Liste mit Zugriffsrechten:

```
acl(Datei 1)= { (Prozess 1, {read, write, own}), (Prozess 2, {append})}  
acl(Datei 2)= { (Prozess 2, {read, write, own} )}
```

- Wenig Speicherbedarf
- Schneller Zugriff
- Schnelles Löschen und Hinzufügen von Objekten

Bei vielen Benutzern kann die Liste lang werden, deshalb:

- Wildcards (*)
- Benutzergruppen

Zugriffskontrollliste (Access Control List oder ACL)

- Weitverbreitete Möglichkeit Zugriffskontrolle zu implementieren
- Basiert auf der Zugriffskontrollmatrix
- Jedes Objekt/Subjekt hat eine eigene Liste mit Zugriffsrechten:

```
acl(Datei 1)= { (Prozess 1, {read, write, own}), (Prozess 2, {append})}  
acl(Datei 2)= { (Prozess 2, {read, write, own} )}
```

- Wenig Speicherbedarf
- Schneller Zugriff
- Schnelles Löschen und Hinzufügen von Objekten

Bei vielen Benutzern kann die Liste lang werden, deshalb:

- Wildcards (*)
- Benutzergruppen

ACL in Windows

- Eingeführt in Windows NT für NTFS Partitionen
- Mögliche Berechtigungen:
 - Vollzugriff
 - Ändern
 - Lesen und Ausführen
 - Ordnerinhalt auflisten (nur bei Ordnern)
 - Lesen
 - Schreiben
 - Spezielle Rechte
- Berechtigungen sind Gruppierungen von 14 speziellen Rechten
- Es gibt Benutzergruppen und eine Gruppe „Jeder“
- Vererbung an Kindelemente bei Windows NT statisch ab Windows 2000 dynamisch
- Neben Dateien in NTFS Partitionen können auch Freigabe, Drucker, Einträge im Active Directory und Registry-Keys geschützt werden

ACL in Windows

- Eingeführt in Windows NT für NTFS Partitionen
- Mögliche Berechtigungen:
 - Vollzugriff
 - Ändern
 - Lesen und Ausführen
 - Ordnerinhalt auflisten (nur bei Ordnern)
 - Lesen
 - Schreiben
 - Spezielle Rechte
- Berechtigungen sind Gruppierungen von 14 speziellen Rechten
- Es gibt Benutzergruppen und eine Gruppe „Jeder“
- Vererbung an Kindelemente bei Windows NT statisch ab Windows 2000 dynamisch
- Neben Dateien in NTFS Partitionen können auch Freigabe, Drucker, Einträge im Active Directory und Registry-Keys geschützt werden

ACL in Windows

- Eingeführt in Windows NT für NTFS Partitionen
- Mögliche Berechtigungen:
 - Vollzugriff
 - Ändern
 - Lesen und Ausführen
 - Ordnerinhalt auflisten (nur bei Ordnern)
 - Lesen
 - Schreiben
 - Spezielle Rechte
- Berechtigungen sind Gruppierungen von 14 speziellen Rechten
- Es gibt Benutzergruppen und eine Gruppe „Jeder“
- Vererbung an Kindelemente bei Windows NT statisch ab Windows 2000 dynamisch
- Neben Dateien in NTFS Partitionen können auch Freigabe, Drucker, Einträge im Active Directory und Registry-Keys geschützt werden

ACL in Linux/POSIX

- Basiert auf POSIX 1003.1e draft 17 und wurden in der Version 2.5.46 in den Kernel eingebracht

Eintrag Typ	Text-Form
Besitzer	user::rwx
Benannter Benutzer	user:name:rwx
Besitzer Gruppe	group::rwx
Benannte Gruppe	group:name:rwx
Maske	mask::rwx
Andere	other::rwx

Tabelle: ACL-Eintrag

- Verzeichnisse besitzen neben der ACL noch eine Default-ACL, die beim Erstellen von Kind-Elementen vererbt wird.
- Die Maske schränkt die Rechte der Gruppen und individuellen Benutzer ein.

ACL in Linux/POSIX

- Basiert auf POSIX 1003.1e draft 17 und wurden in der Version 2.5.46 in den Kernel eingebracht

Eintrag Typ	Text-Form
Besitzer	user::rwx
Benannter Benutzer	user:name:rwx
Besitzer Gruppe	group::rwx
Benannte Gruppe	group:name:rwx
Maske	mask::rwx
Andere	other::rwx

Tabelle: ACL-Eintrag

- Verzeichnisse besitzen neben der ACL noch eine Default-ACL, die beim Erstellen von Kind-Elementen vererbt wird.
- Die Maske schränkt die Rechte der Gruppen und individuellen Benutzer ein.

ACL in Linux/POSIX

- Basiert auf POSIX 1003.1e draft 17 und wurden in der Version 2.5.46 in den Kernel eingebracht

Eintrag Typ	Text-Form
Besitzer	user::rwx
Benannter Benutzer	user:name:rwx
Besitzer Gruppe	group::rwx
Benannte Gruppe	group:name:rwx
Maske	mask::rwx
Andere	other::rwx

Tabelle: ACL-Eintrag

- Verzeichnisse besitzen neben der ACL noch eine Default-ACL, die beim Erstellen von Kind-Elementen vererbt wird.
- Die Maske schränkt die Rechte der Gruppen und individuellen Benutzer ein.

Befähigungen (Capabilities)

Jedes Subjekt besitzt eine Liste von Befähigungen, die aus einem Objekt und den dazugehörigen Rechten besteht.

$$\begin{aligned} \text{cap}(\text{Prozess 1}) &= \{ (\text{Datei 1}, \{ \text{read}, \text{write}, \text{own} \}), (\text{Datei 2}, \{ \text{append} \}) \} \\ \text{cap}(\text{Prozess 2}) &= \{ (\text{Datei 2}, \{ \text{read}, \text{write}, \text{own} \}) \} \end{aligned}$$

- Capability enthält eine Referenz auf das Objekt
- Adressieren von Objekten nicht über die Adresse, ID oder den Namen, sondern über die Capability
- **Problem:** Subjekt darf nicht die Möglichkeit besitzen eigene Befähigungen anzulegen oder zu verändern
- **Lösung:** Tagged Speicher oder Kryptographie

Befähigungen vs. ACL

Der Unterschied zwischen Befähigungen und der ACL wird an zwei wichtigen Fragen deutlich.

- 1 Auf welche Objekte kann ein gegebenes Subjekt zugreifen, und mit welchen Rechten?
- 2 Welche Subjekte können auf ein gegebenes Objekt zugreifen, und mit welchen Rechten?

Aufwand diese Fragen zu beantworten:

	ACL	Befähigungen
Frage 1	hoch	gering
Frage 2	gering	hoch

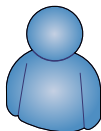
Schlösser und Schlüssel (Locks and Keys)

- Subjekte haben Schlüssel
- Objekte haben Schlösser
- Hat ein Subjekt einen passenden Schlüssel für ein Schloss des Objekts, dann hat es darauf Zugriff.
- Kann beispielsweise kryptographisch implementiert werden

Schlösser und Schlüssel (Locks and Keys)

- Subjekte haben Schlüssel
- Objekte haben Schlösser
- Hat ein Subjekt einen passenden Schlüssel für ein Schloss des Objekts, dann hat es darauf Zugriff.
- Kann beispielsweise kryptographisch implementiert werden

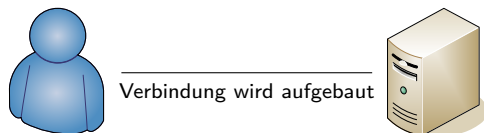
Beispiel:



Schlösser und Schlüssel (Locks and Keys)

- Subjekte haben Schlüssel
- Objekte haben Schlösser
- Hat ein Subjekt einen passenden Schlüssel für ein Schloss des Objekts, dann hat es darauf Zugriff.
- Kann beispielsweise kryptographisch implementiert werden

Beispiel:



Schlösser und Schlüssel (Locks and Keys)

- Subjekte haben Schlüssel
- Objekte haben Schlösser
- Hat ein Subjekt einen passenden Schlüssel für ein Schloss des Objekts, dann hat es darauf Zugriff.
- Kann beispielsweise kryptographisch implementiert werden

Beispiel:



Schlösser und Schlüssel (Locks and Keys)

- Subjekte haben Schlüssel
- Objekte haben Schlösser
- Hat ein Subjekt einen passenden Schlüssel für ein Schloss des Objekts, dann hat es darauf Zugriff.
- Kann beispielsweise kryptographisch implementiert werden

Beispiel:



Server prüft ob empfangene Schlüssel autorisiert ist

Schlösser und Schlüssel (Locks and Keys)

- Subjekte haben Schlüssel
- Objekte haben Schlösser
- Hat ein Subjekt einen passenden Schlüssel für ein Schloss des Objekts, dann hat es darauf Zugriff.
- Kann beispielsweise kryptographisch implementiert werden

Beispiel:



Schlösser und Schlüssel (Locks and Keys)

- Subjekte haben Schlüssel
- Objekte haben Schlösser
- Hat ein Subjekt einen passenden Schlüssel für ein Schloss des Objekts, dann hat es darauf Zugriff.
- Kann beispielsweise kryptographisch implementiert werden

Beispiel:



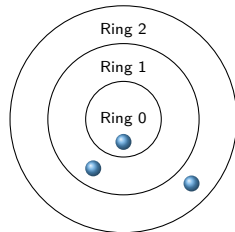
Ringbasierte Zugriffskontrolle (Ring-Based Access Control)

- Der Protected-Mode der Intel-Prozessoren ab 386er besitzt eine Ringbasierte Zugriffskontrolle, die direkt in Hardware implementiert ist
- Der Kernel mit den Treibern läuft auf Ring 0
- Anwendungen laufen auf höheren Ringen
- System-Calls werden durch Gates von der Anwendung zum Kernel geleitet

Ringbasierte Zugriffskontrolle (Ring-Based Access Control)

- Der Protected-Mode der Intel-Prozessoren ab 386er besitzt eine Ringbasierte Zugriffskontrolle, die direkt in Hardware implementiert ist
- Der Kernel mit den Treibern läuft auf Ring 0
- Anwendungen laufen auf höheren Ringen
- System-Calls werden durch Gates von der Anwendung zum Kernel geleitet

Beispiel:

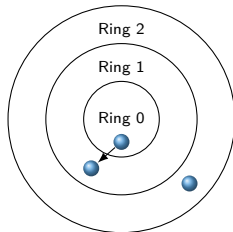


Zugriffe von Ring 0 nach Ring 1 oder Ring 2 sind erlaubt.

Ringbasierte Zugriffskontrolle (Ring-Based Access Control)

- Der Protected-Mode der Intel-Prozessoren ab 386er besitzt eine Ringbasierte Zugriffskontrolle, die direkt in Hardware implementiert ist
- Der Kernel mit den Treibern läuft auf Ring 0
- Anwendungen laufen auf höheren Ringen
- System-Calls werden durch Gates von der Anwendung zum Kernel geleitet

Beispiel:

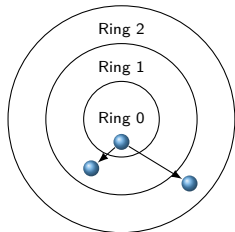


Zugriffe von Ring 0 nach Ring 1 oder Ring 2 sind erlaubt.

Ringbasierte Zugriffskontrolle (Ring-Based Access Control)

- Der Protected-Mode der Intel-Prozessoren ab 386er besitzt eine Ringbasierte Zugriffskontrolle, die direkt in Hardware implementiert ist
- Der Kernel mit den Treibern läuft auf Ring 0
- Anwendungen laufen auf höheren Ringen
- System-Calls werden durch Gates von der Anwendung zum Kernel geleitet

Beispiel:

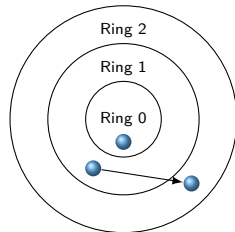


Zugriffe von Ring 0 nach Ring 1 oder Ring 2 sind erlaubt.

Ringbasierte Zugriffskontrolle (Ring-Based Access Control)

- Der Protected-Mode der Intel-Prozessoren ab 386er besitzt eine Ringbasierte Zugriffskontrolle, die direkt in Hardware implementiert ist
- Der Kernel mit den Treibern läuft auf Ring 0
- Anwendungen laufen auf höheren Ringen
- System-Calls werden durch Gates von der Anwendung zum Kernel geleitet

Beispiel:

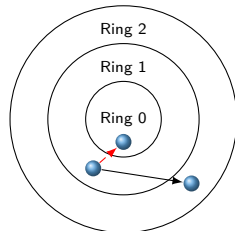


Zugriffe von Ring 1 nach Ring 2
zwar erlaubt.

Ringbasierte Zugriffskontrolle (Ring-Based Access Control)

- Der Protected-Mode der Intel-Prozessoren ab 386er besitzt eine Ringbasierte Zugriffskontrolle, die direkt in Hardware implementiert ist
- Der Kernel mit den Treibern läuft auf Ring 0
- Anwendungen laufen auf höheren Ringen
- System-Calls werden durch Gates von der Anwendung zum Kernel geleitet

Beispiel:

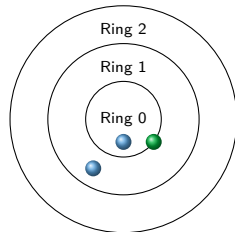


Aber nicht der Zugriff von Ring 1 auf Ring 0.

Ringbasierte Zugriffskontrolle (Ring-Based Access Control)

- Der Protected-Mode der Intel-Prozessoren ab 386er besitzt eine Ringbasierte Zugriffskontrolle, die direkt in Hardware implementiert ist
- Der Kernel mit den Treibern läuft auf Ring 0
- Anwendungen laufen auf höheren Ringen
- System-Calls werden durch Gates von der Anwendung zum Kernel geleitet

Beispiel:

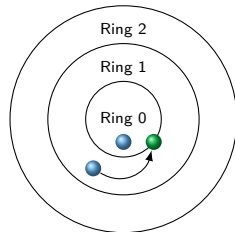


Zugriffe von Ring 1 nach Ring 0 können über geschützte Gates durchgeführt werden.

Ringbasierte Zugriffskontrolle (Ring-Based Access Control)

- Der Protected-Mode der Intel-Prozessoren ab 386er besitzt eine Ringbasierte Zugriffskontrolle, die direkt in Hardware implementiert ist
- Der Kernel mit den Treibern läuft auf Ring 0
- Anwendungen laufen auf höheren Ringen
- System-Calls werden durch Gates von der Anwendung zum Kernel geleitet

Beispiel:

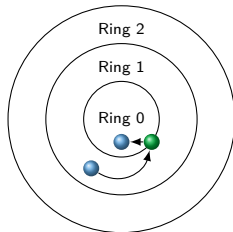


Zugriffe von Ring 1 nach Ring 0 können über geschützte Gates durchgeführt werden.

Ringbasierte Zugriffskontrolle (Ring-Based Access Control)

- Der Protected-Mode der Intel-Prozessoren ab 386er besitzt eine Ringbasierte Zugriffskontrolle, die direkt in Hardware implementiert ist
- Der Kernel mit den Treibern läuft auf Ring 0
- Anwendungen laufen auf höheren Ringen
- System-Calls werden durch Gates von der Anwendung zum Kernel geleitet

Beispiel:



Zugriffe von Ring 1 nach Ring 0 können über geschützte Gates durchgeführt werden.

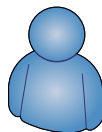
Propagierte Zugriffskontrolle (Propagated Access Control)

- Wenn ein Subjekt ein Objekt liest, wird die PACL (Propagated Access Control List) des Objekts auf das Subjekt übertragen
- Wenn ein Subjekt ein Objekt erzeugt, wird seine PACL auf das Objekt übertragen
- Dieser Mechanismus ist ideal für die ORCON Policy

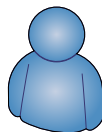
Propagierte Zugriffskontrolle (Propagated Access Control)

- Wenn ein Subjekt ein Objekt liest, wird die PACL (Propagated Access Control List) des Objekts auf das Subjekt übertragen
- Wenn ein Subjekt ein Objekt erzeugt, wird seine PACL auf das Objekt übertragen
- Dieser Mechanismus ist ideal für die ORCON Policy

Beispiel:



Anne



Peter

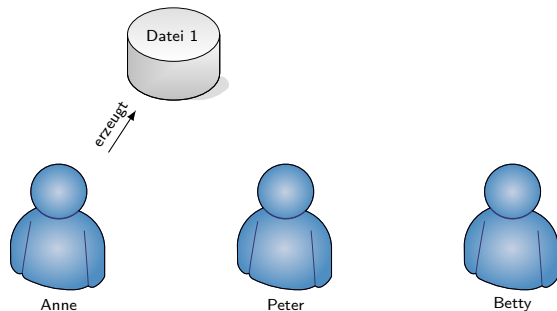


Betty

Propagierte Zugriffskontrolle (Propagated Access Control)

- Wenn ein Subjekt ein Objekt liest, wird die PACL (Propagated Access Control List) des Objekts auf das Subjekt übertragen
- Wenn ein Subjekt ein Objekt erzeugt, wird seine PACL auf das Objekt übertragen
- Dieser Mechanismus ist ideal für die ORCON Policy

Beispiel:

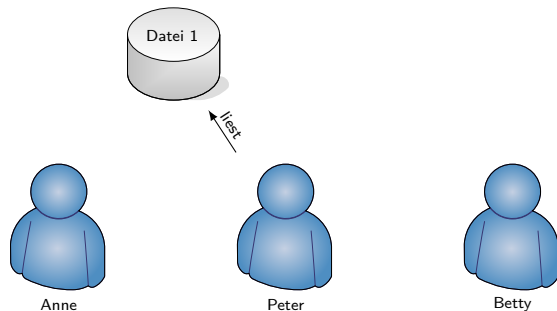


Anne legt die *Datei 1* an und erlaubt Peter den Zugriff auf die Datei.

Propagierte Zugriffskontrolle (Propagated Access Control)

- Wenn ein Subjekt ein Objekt liest, wird die PACL (Propagated Access Control List) des Objekts auf das Subjekt übertragen
- Wenn ein Subjekt ein Objekt erzeugt, wird seine PACL auf das Objekt übertragen
- Dieser Mechanismus ist ideal für die ORCON Policy

Beispiel:

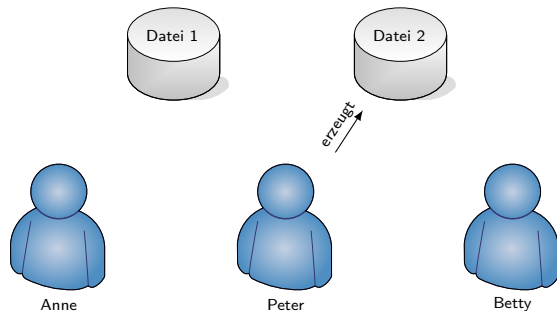


Nachdem Peter aus *Datei 1* liest, ändert sich seine PACL zu
 $PACL(\text{Peter}) = PACL_{\text{Peter}} \cap PACL_{\text{Anne}} = PACL_{\text{Peter, Anne}}$.

Propagierte Zugriffskontrolle (Propagated Access Control)

- Wenn ein Subjekt ein Objekt liest, wird die PACL (Propagated Access Control List) des Objekts auf das Subjekt übertragen
- Wenn ein Subjekt ein Objekt erzeugt, wird seine PACL auf das Objekt übertragen
- Dieser Mechanismus ist ideal für die ORCON Policy

Beispiel:

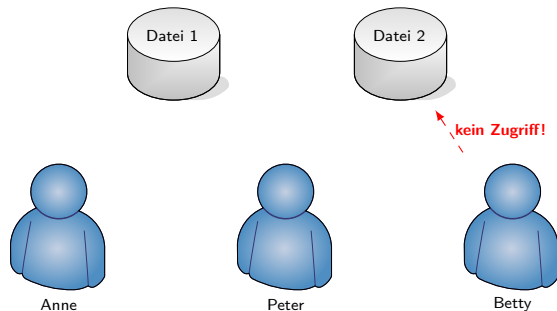


Peter legt die *Datei 2* an mit
 $PACL(Datei 2) =$
 $PACL(Peter) = PACL_{Peter, Anne}$
 $PACL_{Peter}$ erlaubt Betty den
Zugriff auf Dateien.

Propagierte Zugriffskontrolle (Propagated Access Control)

- Wenn ein Subjekt ein Objekt liest, wird die PACL (Propagated Access Control List) des Objekts auf das Subjekt übertragen
- Wenn ein Subjekt ein Objekt erzeugt, wird seine PACL auf das Objekt übertragen
- Dieser Mechanismus ist ideal für die ORCON Policy

Beispiel:



Betty hat keine Berechtigung für die *Datei 2*, da sie nur in $PACL_{Peter}$ steht.

Inhalt

① Das Modell für die Zugriffskontrolle

- Zugriffskontrollmatrix

- Ist ein System sicher?

- Take-Grant Schutzmodell

- Schematic Protection Modell

② Die Mechanismen

- Zugriffskontrollliste (ACL)

 - ACL in Windows

 - ACL in Linux/POSIX

- Befähigungen

- Schlösser und Schlüssel

- Ringbasierte Zugriffskontrolle

- Propagierte Zugriffskontrolle

③ Zusammenfassung

Zusammenfassung

- Zugriffskontrollmatrix-Modell ist ein allgemeines Modell um Zugriffskontrolle zu beschreiben
- Safety-Problem im Allgemeinen nicht entscheidbar
- Es gibt mächtige Modelle, für die das Safety-Problem entscheidbar ist
- ACL als verbreitete Implementierung der Zugriffskontrollmatrix
- Es gibt aber auch andere Mechanismen um Zugriffskontrolle zu ermöglichen

Ende

Vielen Dank für Ihre Aufmerksamkeit!