Skriptum zur Vorlesung

DATENSCHUTZ UND DATENSICHERHEIT

Sicherheitsmodellierung und Zugriffskontrollsysteme

Wintersemester 2001/2002

Vortragender:

Dr. Jörg Pflüger

Sicherheitsmodelle und Zugriffskontrolle

Eine wichtige Anforderung an ein sicheres Systems ist, Daten oder Informationen vor unsachgemäßer Benutzung, unzulässiger Einsicht, Manipulation oder Raub zu bewahren.

Drei sich gegenseitig stützende Technologien sollen die Grundlage für Systemsicherheit schaffen:

- Identifikationsüberprüfung,
- Zugriffskontrolle und
- Zugriffsprotokollierung

Die **Identifikationsüberprüfung** kontrolliert die Identität eines Benutzers, der mit dem System in Kontakt treten will, anhand eines einzigartigen Merkmals.

Die **Zugriffskontrolle** bestimmt, welchem Benutzer Zugriff auf welche Betriebsmittel und Objekte gestattet wird. Die Zugriffskontrolle hat normalerweise eine Identifikationsüberprüfung zur Voraussetzung.

> Es ist wichtig, eine klare Abgrenzung zwischen Identifikationsüberprüfung und Zugriffskontrolle zu treffen. Die Zugriffskontrolle nimmt an, daß die Identität des Benutzers erfolgreich bestätigt worden ist, ihre Wirksamkeit beruht also auf einer korrekten Anwenderauthentifizierung.

Der **Protokollierungsprozeß** (Monitoring) sammelt Daten über die Aktivität im System und analysiert sie, um Sicherheitsübertretungen zu entdecken oder ihre Ursache zu diagnostizieren.

> Zugriffskontrollen sind niemals perfekte Lösungen für das Sichern eines Systems; sie müssen mit Protokollierung verbunden werden.
>
> Protokollierung verlangt die Aufnahme aller Benutzeranforderungen und Aktivitäten für ihre (spätere) Analyse. Die Analysen können im off-line oder im on-line Betrieb vorgenommen werden. Es wird analysiert, wie Benutzer das System verwenden, um Hinweise auf Benutzer zu finden, die ihre "Berechtigung" mißbräuchlich verwenden. Protokollierung kann auch mögliche Fehler im Sicherheitssystem zu Tage fördern.

Authentifizierung

Identifikationsüberprüfung ist die Hauptsicherungsanwendung, von der andere Sicherheitsanwendungen abhängen. Ohne gute Identifikationsüberprüfung ist keine wirkungsvolle Zugriffskontrolle oder Einbruchsicherung möglich.

Es gibt verschiedene Möglichkeiten, Benutzer eindeutig zu identifizieren. Methoden sind beispielsweise

- Paßwortauthentifikation (login/password-Konzept),
- Token-basierte Identifikation (Hardware),
- biometrische Identifikationsüberprüfung
- starke Authentifizierung etc.

Identifikationsüberprüfung wird im allgemeinen mit "Benutzer authentifiziert sich für eine Maschine" assoziiert. In einer vernetzten Landschaft wird die Identifikation schwieriger, denn hier wird eine "Computer zu Computer" Überprüfung nötig. Ein Nichtberechtigter, der den Netzwerkverkehr beobachtet, kann z.B. die Identifikationsüberprüfungsprotokolle eines legitimen Benutzers wiederholen (Replay-Angriff bei Paßworteingabe -> starke Authentifizierung).

Im allgemeinen führt eine Identifikationsüberprüfung die Identität unidirekt von einem Menschen/ Computer zu einem anderen Computer. Oft wird Überprüfung in beiden Richtungen verlangt. Gegenseitige Identifikationsüberprüfung ist z.B. in einer Client-Server-Situation nützlich. Die Identifikationsüberprüfung einer Maschine gegenüber einem Benutzer ist auch sinnvoll, um zu verhindern, daß sich Benutzer an nicht vertrauenswürdigen Rechnern anmelden (Trojanische Pferde).

Zugriffskontrollsysteme

Unter Zugriffskontrolle versteht man die Art und Weise der Rechtevergabe und Rechteverwaltung des Zugriffs von Benutzern (Subjekte) oder Prozessen auf Betriebsmittel (Objekte, Prozesse) eines Systems.

Gestaltungs-Grundsätze

1. minimale vs. maximale Zugriffsrechte (need to know vs. maximized sharing principle)
2. offene vs. geschlossene Systeme
3. zentralisierte vs. dezentralisierte Kontrolle
4. Privilegienfortpflanzung (obligatorische vs. Ermessenssysteme)

1. Minimale Zugriffsrechte vs. maximale Beteiligung

"Need-to-know principle":

Jedes Subjekt muss mit dem Minimum an Information auskommen, das für seine Arbeit notwendig ist.
Dies setzt voraus, dass es möglich ist, den Umfang an Informationen zu bestimmen, die ein Benutzer brauchen wird, was in der Praxis meist schwierig ist.

"Maximized-sharing principle":

Hat die Zugänglichkeit zu größeren Informationsmengen höhere Priorität als das Schutzmotiv, (unter bestimmten Einschränkungen) kann der Benutzer durch Programm- und Datensharing Zutritt zur maximal möglichen Informationsmenge gewinnen.

2. Offene vs. geschlossene Systeme

In *geschlossenen Systemen* sind nur Zugriffe möglich, die explizit erlaubt sind,;

bei *offenen Systemen* sind alle Zugriffe möglich, die nicht explizit verboten wurden.

3. Zentrale vs. dezentrale Kontrolle

Für das System ist es wichtig, festzustellen, wer Zugriffsrechte vergeben bzw. zurücknehmen kann.

Gibt es nur eine einzige verantwortliche Stelle, liegt ein **zentralisierte Autorisierung** vor (z.B.: zentralisierte Datenbanken - Datenbankadministrator).

Dezentralisierte Autorisierung ist für verteilte Systeme typisch, wo das Eigentums-Privileg die vorhandenen Ressourcen bestimmten Benutzern zuordnet (z.B.: in Netzwerken häufig angewendet).

4. Privilegienfortpflanzung

Man spricht von einer **obligatorischen oder verbindlichen Sicherheitspolitik** ("mandatory policy" oder "mandatory systems"), wenn einmal vergebene Zugriffsrechte nur von der Autorisierungsperson modifiziert werden können. Die Autoritätsperson hat vollständige Kontrolle über alle Privilegien.

Verbindliche Sicherheitsmodelle erlauben den Zugriff auf Objekte anhand einer zentral festgelegten Klassifikation von Objekten und Subjekten im System. Es werden Zugriffsklassen für Objekte und Subjekte definiert, die den Sicherheitscharakter der Objekte bzw. Subjekte darstellen. Der Zugriff auf Objekte wird nur dann gewährt, wenn der Zugriffsmodus der Klassifikation des Benutzers und des. Objekts in einem zulässigen Verhältnis stehen.

In **diskreten oder Ermessensystemen** können von den Benutzern (Objekteigentümer, Erzeuger) die Zugriffsrechte auf "ihre" Objekte an andere Benutzer weitergegeben werden ("discretionary policy" oder "discretionary systems").

Daraus ergeben sich Probleme eines ungesicherten Informationsflusses und mangelnde Übersicht über die Rechtevergabe..

Gruppierung von Subjekten und Objekten

Durch eine Gruppen- oder Klasseneinteilung von Subjekten oder Objekten werden die Zugriffsspezifikation und das Zugriffsmanagement erleichtert.

Beim Entwurf des Systems muß das Management der Gruppe analysieren, welche Folgen daraus resultieren, daß ein Benutzer zu verschiedenen Gruppen gehört. Auf der operationellen Ebene muß festgestellt werden, wer über die Zuständigkeit entscheidet.

In manchen Systemen werden die Gruppen mittels Defaultwerten implementiert.

In Mehrebenensystemen stellt die Klassifikation einen automatischen Gruppenmechanismus dar.

Beispiel: UNIX mit Owner, Group, Others

(Theoretisch) einfache Mechanismen der Zugriffüberwachung

Zugriffsmatrix

In den Sicherheitsanforderungen ist festgelegt, **wer, wann** und **wie** auf bestimmte Datenobjekte zugreifen darf. Die geläufigsten Beziehungen sind dabei sicherlich die zwischen dem **Rechner**, den auf dem Rechner enthaltenen **Dateien** und dem **Benutzer**.

Selbstverständlich gibt es auch noch eine Vielzahl anderer Subjekte, für die Zugriffsrechte vergeben werden können und müssen. Dies können zum Beispiel andere Rechner in verteilten Systemen, oder auch verschiedene Prozesse des eigenen oder fremden Rechners sein.

Auch die Art der zu schützenden Objekte ist unterschiedlich: Es können die gesamte Festplatte, einzelne Verzeichnisse, Dateien oder nur ganz bestimmte Datensätze gesichert werden.

Subjekte sind **aktive Elemente** (Benutzer, Prozesse) die Zugriff zu Objekten suchen.

Objekte wiederum sind "**passive" Elemente** (Hardware, Dateien oder Prozesse).

Das **Zugriffsrecht** ist die Erlaubnis oder Fähigkeit, eine bestimmte Operation auf einem bestimmten Objekt auszuführen.

Zugriffsbeziehungen lassen sich sehr einfach in Zugriffsmatrizen darstellen.

Zugriffssrecht	**Objekt1**	**Objekt2**	**Objekt3**	**....**
Subjekt1	lesen	schreiben	--	...
Subjekt2	schreiben	--	lesen	....
.....	lesen	--	....	....

Bedingt durch die dünne Besetzung und die dynamische Veränderung einer solchen Matrix, eignet sich diese nur für den Einsatz in kleinen Umgebungen. In der Praxis werden daher häufiger

Zugriffskontrollisten ("access control lists") und
Capability-Listen ("capability lists")

verwendet.

Access Control Lists

Hierbei wird für jedes Objekt eine Liste geführt, die angibt, wer in welcher Form auf es zugreifen darf. Benutzer, die nicht in der Liste enthalten sind, erhalten ein voreingestelltes (default) Zugriffsrecht.
(Entspricht den Spalten der Zugriffsmatrix, wobei nur die besetzten Felder aufgeführt werden.)

Objekt i	Subjekt j_1	Zugriffsrechte	Subjekt j_2	Zugriffsrechte	.....

Dieses Verfahren erspart gegenüber der Zugriffsmatrix viel Platz.

Zugriffsrechte für einzelne Objekte sind leicht veränderbar. Subjekte können einfach gelöscht, Zugriffsrechte geändert werden.
Die Gesamtrechte eines Subjektes sind allerdings schwer erkennbar und nicht einfach zu ändern.

Bei größeren Systemen stößt dieses Verfahren allerdings auch schnell an seine Grenzen.
Dann können noch zusätzliche Strukturen wie z.B. Gruppen eingeführt werden. Hierbei werden Benutzer bestimmten Gruppen zugeordnet, die dann wieder default-Zugriffsrechte haben. Für diesen Fall muß jedoch eindeutig definiert sein, wie sich das System bei Überschneidungen zu verhalten hat, ob also Benutzerrecht vor Gruppenrecht gilt oder umgekehrt. Auch die Vereinigung aus Benutzer- und Gruppenrecht oder der Durchschnitt wären Möglichkeiten, diese Fälle zu behandeln.

Capability Lists

Capabilities kann man sich wie Eintrittskarten vorstellen. Man erwirbt das Recht auf den Zugriff auf bestimmte Dateien, Prozesse etc. Für jedes Subjekt wird also eine Liste der Objekte gespeichert, auf die es Zugriff hat, zusammen mit den jeweiligen Zugriffsrechten. Objekte, auf die das Subjekt keine Zugriffsrechte besitzt, werden nicht angegeben. (Entspricht den Zeilen der Zugriffsmatrix.)

Subjekt i	Objekt j_1	Zugriffsrechte	Objekt j_2	Zugriffsrechte	.....

Besitzt ein Subjekt eine Capability für ein bestimmtes Objekt, kann es die spezifizierten Operationen auf dem Objekt ausführen.

Hier ist es natürlich leicht die Rechte eines Nutzers zu erkennen, aber schwer dies für Objekte zu überprüfen (problematisch bei eventuellem Mißbrauch)-

Die Capabilities werden von einer Zentralen Verwaltung ausgegeben. Diese weiß zwar in der Regel an wen, kann allerdings die Weitergabe nicht mehr kontrollieren. Beim Einlösen der Capability wird nämlich nicht die Identität des Einlösenden festgestellt. Es besteht die Gefahr der Duplizierung einer Capability.

Die Capability kann zwar auch so hergestellt werden, daß sie nur für eine bestimmte Dienstleistung gültig ist und danach ihren Wert verliert. Ein solches Verfahren ist aber sehr aufwendig zu implementieren.

Es wäre angebracht, die Capabilities nicht in Dateien, sondern in durch den Betriebssystemkern geschützten Datenfeldern aufzubewahren. Die Erzeugung und Verteilung wird dann von diesem überwacht. Bei verteilten Systemen muß dann natürlich überall ein solcher Betriebssystemkern existieren. Ein Problem entsteht dann, wenn eine Capability nachträglich als ungültig erklärt werden soll.

Zugriffsprotokollierung

Die Aufzeichnung und Analyse der Historie von Ereignissen in einem System ist eine notwendige Maßnahme, um Sicherheitsübertretungen oder entsprechende Versuche zu bestimmen. Zugriffsprotokollierung erfordert die Registrierung von Benutzern und das Protokollieren von deren Aktivitäten für spätere Prüfungen und möglicherweise als Beweismittel.

Je nach Aktivität im System können diese Protokolle sehr schnell sehr groß werden. Nach Möglichkeit sollten automatisierte Werkzeuge verwendet werden, um Daten zu strukturieren, zu komprimieren und zu reduzieren. Dieser Datenreduktionsprozeß kann zum Beispiel kurze Zusammenfassungen von Benutzer-verhalten, anormalen Ereignissen oder Sicherheitsvorfällen liefern. Der Revisor (Administrator) kann dann die Zusammenfassung der relevanten Ereignisse auswerten, ohne jedes einzelne Ereignis zu prüfen.

Sogenannte "intrusion detection systems" sind Werkzeuge, die eine Analyse des Benutzerverhaltens automatisieren sollen, um Sicherheitübertretungen ohne menschliche Unterstützung zu erkennen.

Intrusion Detection Systeme können in passive oder aktive Varianten unterteilt werden.

> *Passive Systeme*, die im allgemeinen off-line laufen, analysieren die Protokolldaten und benachrichtigen den Revisor bei potentiellen Störungen oder Übertretungen, der dann geeignete Maßnahmen ergreift.

Aktive Systeme analysieren in Echtzeit Protokolldaten. Neben der Benachrichtigung des Revisors bei ungewöhnlichen Vorfällen können diese Systeme selbst Schutzmaßnahmen ergreifen. Diese beinhalten unter anderem verdächtigte Prozesse zu "killen", Benutzer auszuschalten, Privilegien zu entziehen oder auch Benutzer-Accounts zu sperren.

Es existieren unterschiedliche Ansätze zur Implementierung von Übertretungsentdeckungssystemen (z.B. auch nach dem Vorbild von Immun-Systemen).

Kein einziger Ansatz kann zufriedenstellend die vielfältigen Arten von Übertretungen und das Unterlaufen von Sicherheitsbarrieren erkennen, sondern detektiert nur eine bestimmte Teilmenge.

Übertretungsmöglichkeiten komplexer Sicherheitssysteme können (noch?) nicht in gewünschter Qualität eingegrenzt und ein zuverlässiger Schutz automatisiert werden.

Entwicklung von Sicherheitssystemen

Zum Schutz von Daten, Programmen, Rechnern und Übertragungsleitungen vor vielfältigen Bedrohungen, ist es notwendig, in einem Computersystem Mechanismen zu haben, die diesen Gefahren entgegenwirken.

Die jeweilige Sicherheitspolitik, die in Grundsätzen festgelegt wird, führt zu Sicherheitsanforderungen, aus denen die Spezifikation eines Systemssystems abgeleitet werden sollte. Die Erfordernisse werden in Sicherheitsregeln ausgedrückt, die im System gespeichert und überprüft werden.

In der Modellierungs- und Implementierungsphase sollte dann die Korrektheit und Vollständigkeit des Zielsystems bezüglich der Grundsätze und Erfordernisse sichergestellt werden. Die Modellierung und das zugehörige Modell sind daher Teil der Entwicklung eines sicheren Systems ("secure systems process").

Komponenten des Entwicklungsprozesses:

- Artikulation von Sicherheitsgrundsätzen und Erfordernissen auf nicht-formale Weise
- Aufstellung eines formalen Sicherheitsmodells
- Formale Spezifikation auf Basis des Modells
- Implementierung des Systems
  - Implementierung der Sicherheitsprozeduren
  - Tests
  - Formale Überprüfung dieser Prozeduren

Ein **Sicherheitsmodell** ist ein, auf hohem Abstraktionsniveau stehendes, softwareunabhängiges, konzeptionelles Modell, das die Sicherheitsanforderungen eines Systems definiert und spezifiziert. Es wird aufgestellt mit dem Ziel, sicherheitsrelevante Eigenschaften des Systems zu beschreiben und einer formalen Analyse zugänglich zu machen.

Eine formale Analyse ist/wäre beim Nachweis von Sicherheitseigenschaften wichtiger als eine (ohnehin kaum mögliche) formale Verifikation der Korrektheit von Softwaresystemen, weil wir hier nicht nur mit unvorhergesehenem ("ungeschicktem"), sondern mit böswilligem Verhalten der Nutzer zu rechnen haben. Man kann nicht auf das gutmütige Verhalten der Benutzer bauen, die auf bekannt gewordene Fehler mit einem Workaround reagieren, sondern Schwachstellen werden genutzt, um ein Schutzsystem gänzlich zu unterlaufen

Die Erstellung von Sicherheitsmodellen ist daher auch selten trivial und wird selten konsequent durchgeführt. (*Milnet* ist bei Hackern bekannt dafür, daß die meisten DoD-Rechner leicht zu knacken sind ;-)

Dies kann aber keine Entschuldigung für die sträfliche Vernachlässigung von einfachen Sicherheitsanforderungen bei der überwiegenden Zahl kommerzieller Software sein.

Prinzipien von Mehrebenensysteme

Die Systemobjekte werden zu Klassen zusammengefaßt.

Jedem Objekt wird eine **Sicherheitsklasse** (security class - **SC = (Cl,Cat)**) zugeordnet, die aus einem

Klassifikationsniveau (classification level - Cl) und einer

Kategorienmenge (set of categories - Cat) besteht.

Typische Klassifikationsniveaus sind etwa:

Streng geheim
Geheim
Vertraulich
Frei zugänglich

Die Kategorien sind Sicherheitsbereichen oder **Sektoren** zugeordnet.

In militärischen Anwendungen könnten die Bereiche etwa "Nuklear", "Marine" oder "Aufklärung" heißen, in der Geschäftswelt "Produktion", "Management" oder "Verwaltung".

Bezeichnet m die Zahl der Systemgebiete oder Sektoren, ist die Zahl der prinzipiell abzuleitenden Kategorien 2^m.

Für die Sicherheitsklassen existiert eine Halbordnung "≤":

Für je zwei Klassen (Cl,Cat) und (Cl',Cat') gilt

(Cl, Cat) ≤ (Cl', Cat') *gdw* Cat ist in Cat' enthalten und Cl niedriger-gleich Cl'.

Die Sicherheitsklasse (Cl', Cat') "dominiert" also die Sicherheitsklasse (Cl, Cat), wenn die erstgenannte Klasse das gleiche oder ein höheres Sicherheitsniveau hat und die Menge der Sicherheitskategorien eine Obermenge der Kategorienmenge der zweitgenannten Sicherheitsklasse ist.

Beispiele: (2, {Nuklear}) < (2, {Nuklear, Nato})

(2, {Nuklear}) < (3, {Nuklear})

Die Systemsubjekte, üblicherweise die Benutzer, werden je nach ihrer Rolle in **Unbedenklichkeitsstufen** eingeteilt, die den obigen Sicherheitsklassen ähnlich sind;

beim Militär: "clearance" Stufen;

in kommerziellen Systemen: Datenbankadministrator, Systemanalytiker, Programmierer.

Es werden Regeln aufgestellt, die den Zugriff der Subjekte zu den Objekten beschreiben.

Im Bell-LaPaluda.Modell etwa werden diese Regeln in axiomatischer Form angegeben.

Informationsflußkontrolle

Keinen ausreichenden Schutz bieten die meisten Sicherheitsmodelle (Mehrebenensysteme) bei der Informationsflußkontrolle.

Beispiel:

> Benutzer A möchte an Benutzer B eine Datei weitergeben, dabei jedoch sichergehen, daß Benutzer C die Datei nicht erhält. Natürlich kann er den Zugriff von Benutzer C auf seine Datei verbieten. Wenn aber Benutzer B eine Kopie und an Benutzer C weitergibt, kann A nichts dagegen tun.

Diese Problematik kam zuerst im militärischen Bereich auf. Die Kontrolle des Informationsflusses kann aber auch kompliziertere Formen (z. B. bei unterschiedlichen Sicherheitsstufen) annehmen.

> Eine "verräterische" Person mit hoher Sicherheitsstufe, die ihre Sicherheitsstufe dynamisch reduzieren darf, kann streng geheime Informationen in solche mit niedrigerer Geheimhaltungsstufe umschreiben, wodurch diese Personen mit niedriger Sicherheitsstufe zugänglich gemacht werden.

> Bei der Weitergabe von Informationen niedriger Geheimhaltungsstufe an Personen mit höherer Geheimhaltungsstufe können eventuell Trojanische Pferde mitgeliefert werden: Ein niedrig eingestufter Benutzer hat ein nützliches Programm geschrieben. Wenn es von einem hoch eingestuften Benutzer aufgerufen wird, liest es dessen Dateien und stuft sie niedriger ein, so daß der Programmautor sie lesen kann.
>
> Um diese Möglichkeit zu verhindern, werden im Bell-LaPadula System Regeln aufgestellt, die festlegen, daß "processes can read down and write up, but never vice versa":

Außerdem gibt es das schwer in den Griff zu bekommende Problem des Informationsflusses durch “hidden channels” (z.B. “Kodieren” durch Größen oder Zeitintervalle).

Das Bell-LaPaluda Modell

Das Bell-LaPaluda Modell bezieht sich auf den Schutz von Mehrebenensystemen (Betriebs-systemelemente). Jedem Systemsubjekt/Systemobjekt wird eine Sicherheitsstufe zugeordnet, und es werden Sicherheitsaxiome angegeben, gemäß denen die Sicherheitseigenschaften überprüft werden.

(Das Bell-LaPadula Modell ist eine formale Darstellung der einfachen Tatsache, daß ein General mehr wissen und weniger erzählen darf als ein einfacher Soldat ;-)

Modellelemente

Subjekte: Benutzer bzw. Prozesse

Objekte: Elemente, die Information enthalten (Files, Speicherbereiche, Prozesse etc.)

Sicherheitsklassen: Klassifikationsniveau und Kategorienmenge

Streng geheim (Top Secret)
Geheim (Secret)
Vertraulich (Confidential)
Frei zugänglich (Unclassified)

mögliche Subjekt- und Objektklassifikationsniveaus

Die Kategorien variieren nach Anwendungsfeld (z.B. Nuklearbereich, Nato, Navy in militärischen Systemen).

Die Sicherheitsklassen sind teilgeordnet, daher gilt z. B. (wenn L Klassen, Cl Niveaus und Cat Kategorien bezeichnet), daß

$$L1 = (Cl1, Cat1) \geq L2 = (Cl2, Cat2)$$

impliziert, daß das Sicherheitsniveau Cl1 höher oder gleich dem von Cl2 ist und Cat1 eine Obermenge von Cat2 darstellt.

Zugriffszustand: Menge von vier Komponenten (b, M, f, H), wobei b den laufenden Zugriff ausdrückt, der aus einer Menge mit drei Elementen besteht: (Subjekt, Objekt, Zugriffsart).

Die Zugriffsarten zeigt die folgende Tabelle.

Zugriffsart	**Bedeutung**
e	Ausführen (execute)
r	Lesen (read) ohne Veränderung
a	Hinzufügen (append = change with no read)
w	Schreiben (write = read and change)

Zugriffsmatrix M: enthält alle Zugriffsarten auf ein Objekt.

Sicherheitsstufenfunktion f: berechnet die Sicherheitsstufen der Subjekte/ Objekte. Die Subjekte besitzen eine maximale Sicherheitsstufe und eine gegenwärtige Sicherheitsstufe; Objekte besitzen immer nur eine Stufe.

Hierarchie H: wird durch einen Baum aller Objekte beschrieben, auf die Zugriff besteht. Dadurch werden Objekte mit ihren Erzeugern (Elterngeneration) und deren Nachkommen (Kindergeneration) in Beziehung gesetzt. In Hierarchien dominiert das Sicherheitsniveau eines Objekts das Niveau der Erzeugergeneration.

Operationen:

Das Modell beruht auf 11 Operationen, die in die folgenden Gruppen unterteilt werden:

Veränderung des laufenden Zugriffs, z.B. *get/release access*

Veränderung des garantierten Zugriffs, z.B. *give/revoke access (setzt w voraus)*

Veränderung der Hierarchie, z.B. *create/delete object ((De-)Aktivierung)*

Veränderung der Sicherheitsstufenfunktion, z.B. *change subject current security level,*
change object security level (nur erhöhen),
beides im Rahmen des Sicherheitsniveaus des Subjekts

Axiome:

Operationen werden nur erlaubt, wenn sie folgenden Eigenschaften genügen:

einfache Sicherheitseigenschaft:

Ist ein Subjekt S mit einem Sicherheitsniveau "L-subj" ausgestattet und ein Objekt O mit einem Sicherheitsniveau "L-obj", darf das Subjekt S das Objekt genau dann lesen oder ausführen, wenn L-obj <= L-subj gilt, also das Sicherheitsniveau des Subjekts höher oder gleich dem des Objekts ist.

**- Eigenschaft (star property)*:

S darf die Zugriffsart "append" am Objekt O dann und nur dann ausführen, wenn L-obj >= L-subj gilt.

S darf die Zugriffsart "write" am Objekt O dann und nur dann ausführen, wenn L-obj = L-subj gilt.

S darf die Zugriffsart "read" am Objekt O dann und nur dann ausführen, wenn L-obj <= L-subj gilt.

Dies soll verhindern, daß sensitive Informationen in Objekte einer niedrigeren Sicherheitsstufe kopiert werden, d.h. "processes can read down and write up, but never vice versa" (Ross Anderson).

Alle laufenden Zugriffe werden in einer Zugriffsmatrix dargestellt.

Die stärkste Eigenschaft des Modells ist seine Einfachheit. Jedoch bleibt ein (schon angesprochenes) Sicherheitsproblem ungelöst: Ein Subjekt mit hoher Sicherheitsstufe, das seine Sicherheitsstufe dynamisch reduzieren kann, kann Informationen hoher Sicherheitsstufe in niedrigere Stufen schreiben, wodurch geheime Informationen Subjekten mit niedriger Sicherheitsstufe zugänglich gemacht werden können.

(+ Probleme mit “hidden channels”)