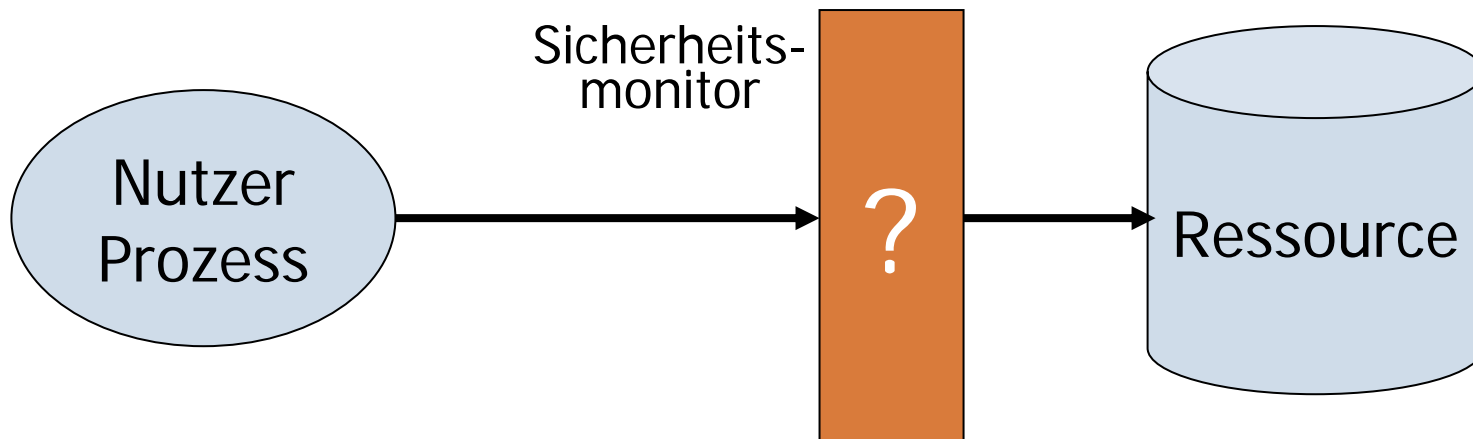


Sicherheit in Betriebssystemen

Prof. Dr.-Ing. Torben Weis
University Duisburg-Essen

Zugriffsschutz

- Annahmen
 - Das BS weiß, wer zugreifen möchte
 - Der Nutzer hat sich mit Name und Passwort oder durch eine Challenge/Response ausgewiesen
 - Zugriff erfolgt über einen Sicherheitsmonitor
 - Das BS muss verhindern, dass man an dem Monitor vorbeikommt



Access Control Matrix

Objects

	File 1	File 2	File 3	...	File n
User 1	read	write	-	-	read
User 2	write	write	write	-	-
User 3	-	-	-	read	read
...					
User m	read	write	read	write	read

Subjects

Varianten der Implementierung

- Access control list (ACL)
 - Speichert die Spalten der Matrix bei den Ressourcen
- Capability
 - Der Anwender erhält ein Ticket für eine Ressource
 - Zwei Untervarianten
 - BS speichert intern die Zeilen bei den Nutzerdaten
 - Der Nutzerprozess hält ein Ticket in einem **fälschungssicheren Speicherbereich**

	File 1	File 2	...
User 1	read	write	-
User 2	write	write	-
User 3	-	-	read
...			
User m	read	write	write

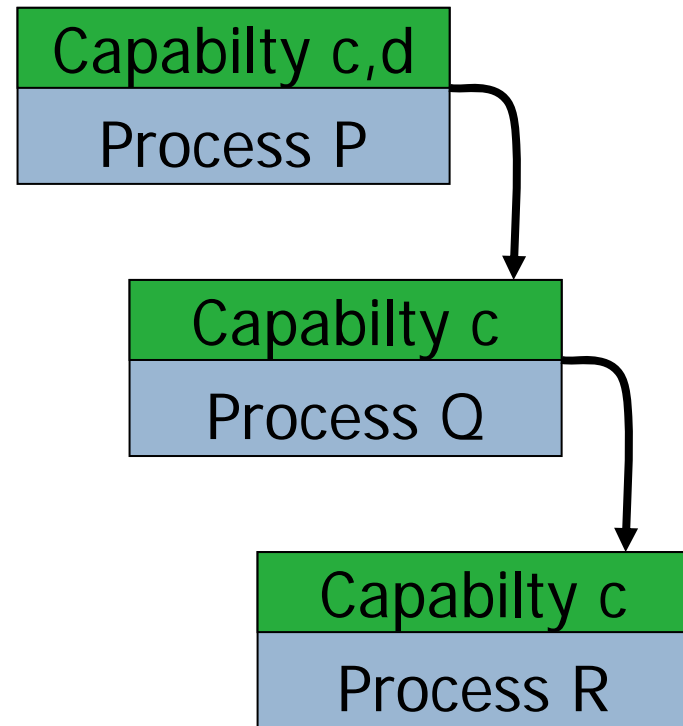
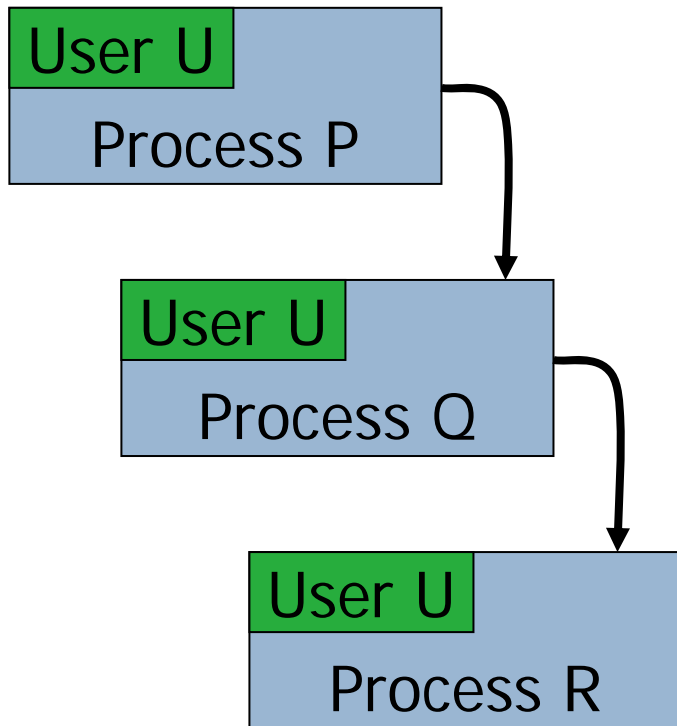
Capabilities

- Ewiges Zukunftskonzept
- Examples
 - Dennis and van Horn, MIT PDP-1 Timesharing
 - Hydra, StarOS, Intel iAPX 432, Eros, ...
 - Amoeba: Verteilte fälschungssichere Tickets
- References
 - Henry Levy, Capability-based Computer Systems
<http://www.cs.washington.edu/homes/levy/capabook/>
 - Tanenbaum, Amoeba papers

ACL vs Capabilities (1)

- Access Control List
 - Liste ist mit der Ressource verbunden
 - Der Nutzer wird in der Liste gesucht
 - Der Nutzer muss dem System bekannt sein
- Capabilities
 - Capability ist ein fälschungssicheres Ticket
 - Tickets können zwischen Prozessen ausgetauscht werden
 - Sicherheitsmonitor überprüft das Ticket
 - Der Sicherheitsmonitor muss nicht die Identität des Tickets kennen

ACL vs Capabilities (2)



ACL vs Capabilities (3)

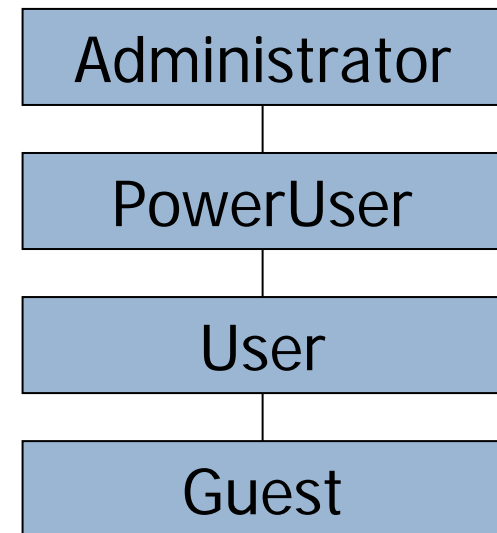
- Delegation
 - Cap: Prozess kann Capability zur Laufzeit weiterreichen
 - ACL: ?????
- Revocation
 - ACL: Nutzer wird aus der ACL entfernt
 - Cap: Wie woll das BS einem Prozess ein Ticket entziehen?
 - Das BS muss wissen, wo Capabilities gespeichert sind
 - Das BS muss jede Kopie (und daher jedes Lesen) eine Capability überwachen
 - Das nimmt einen Großteil der Flexibilität, die man ursprünglich haben wollte
 - Pragmatisch: Capabilities sind zeitlich beschränkt

ACL vs Capabilities (4)

- Capabilities sind wie Universitätschlüssel
- Eine Capability kann man einfach weitergeben
 - z.B. die Fähigkeit, eine Tür zu öffnen
 - Einfach den Schlüssel weitergeben
- Problematisch ist der Diebstahl eines Schlüssels
 - Alle Schlösser müssen ausgetauscht werden
- Wie verhindert man das unerlaubte Kopieren von Schlüsseln?

Rollen / Gruppen

- Nutzer können Rollen einnehmen
 - Administrator, PowerUser, User, Guest
 - Zugriffsrechte werden an Rollen vergeben – und damit indirekt an Nutzer
- Hierarchie von Rollen
 - Partielle Ordnung der Rollen
 - Jede Rolle erbt die Zugriffsrechte der unteren Rolle
 - Für jede Rolle müssen daher nur zusätzliche Zugriffsrechte vermerkt werden



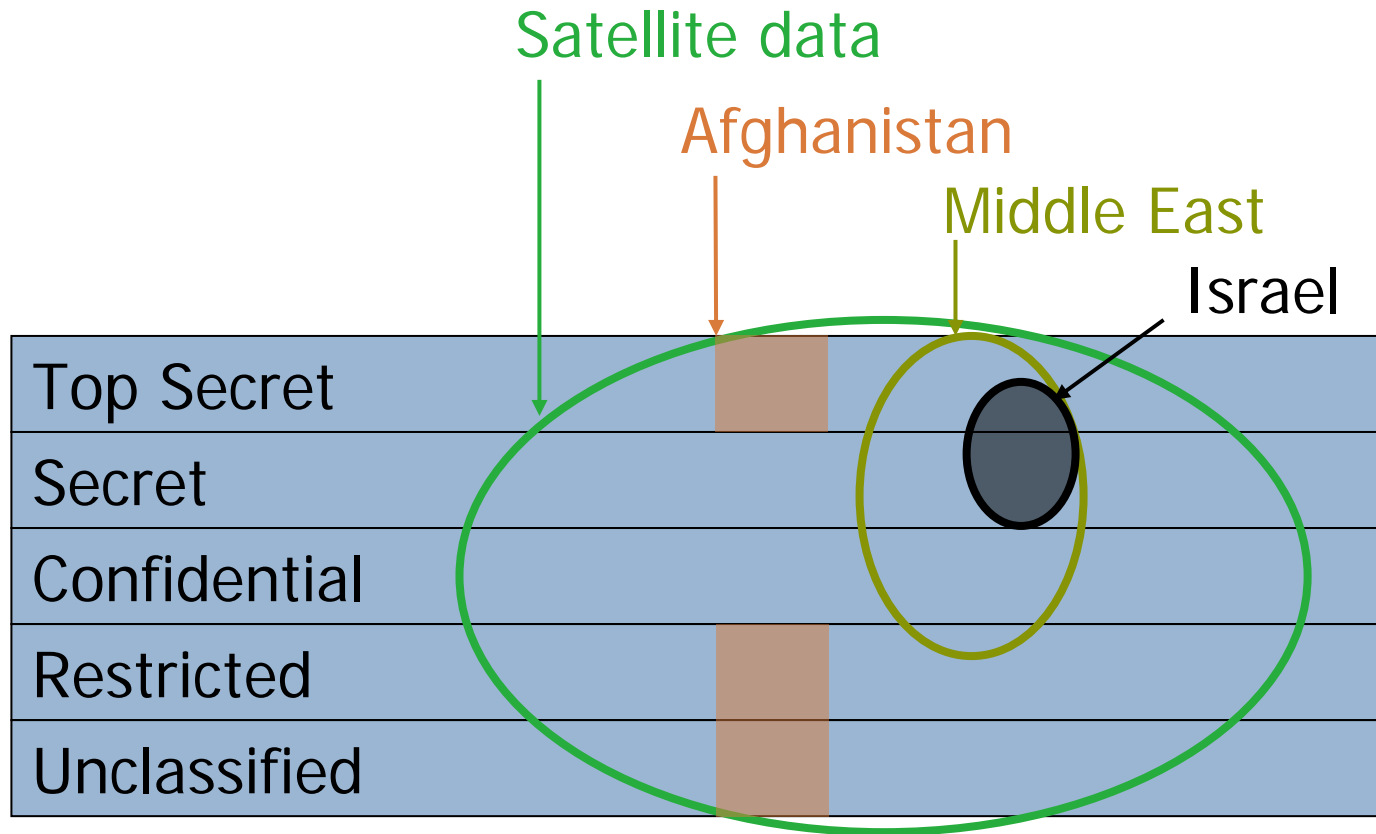
Zugriffsrecht

- Zugriffsrecht = $\langle \text{Operation, Ressource} \rangle$
- Hierarchie von Zugriffsrechten
 - User hat Recht r , und $r > s \rightarrow$ User hat Recht s
 - Beispiel: Der User hat Leserecht auf ein Verzeichnis \rightarrow er hat Leserecht auf alle Dateien des Verzeichnisses
- Hauptproblem der Zugriffskontrolle
 - Komplexe Mechanismen brauchen komplexe Konfigurationen
 - Administration wird sehr aufwendig
 - Rollen, oder andere Organisationsprinzipien versuchen das Problem zu lösen

Military Security (1)

- Multi-Level Security
 - 2-dimensionales Schema:
Classification (top secret, secret, ..., public)
Compartment
- Jedes Dokument gehört zu ...
 - Einem oder mehreren Compartments
 - Und hat eine Classification
- Beispiele
 - (Secret, {Isreal})
 - (Top Secret, {Isreal, Nuclear})

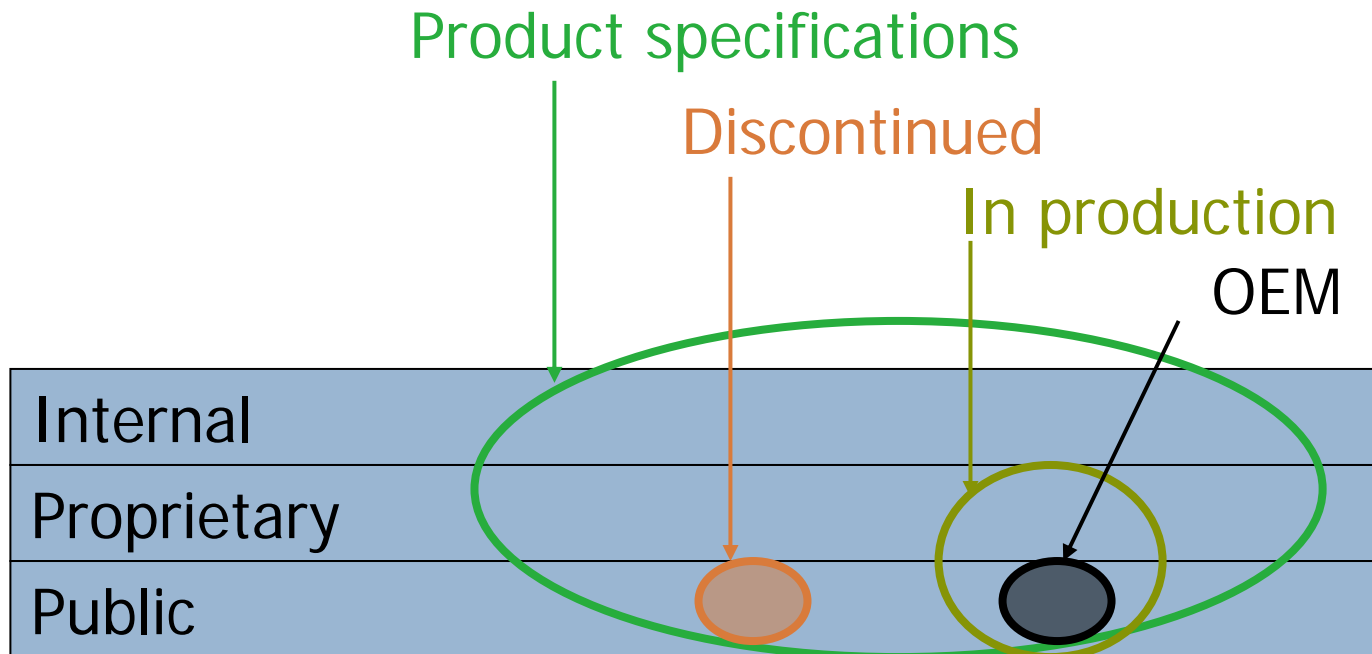
Military Security (2)



Military Security (3)

- Clearance
 - Jede Person hat eine Clearance
 - Eine Clearance hat folgende Form:
(classification, {compartment₁, ..., compartment_n})
- Access Control
 - Dokument X ist geheim: (class_X, compartments_X)
 - Person P hat eine Clearance: (class_P, compartments_P)
 - If $C(X) \leq C(P) \Rightarrow P$ darf auf X zugreifen
- Der \leq Operator
 - $C(X) \leq C(P) \Leftrightarrow$
class_X \leq class_P und
compartments_X \subseteq compartments_P

Kommerzielle Version



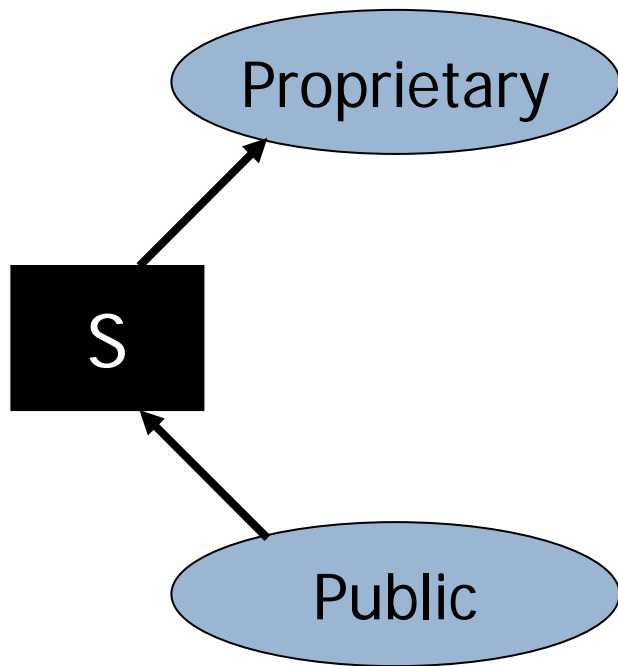
Bell–LaPadula Confidentiality Model (1)

- Wann darf man Daten lesen?
- Zwei Eigenschaften (mit komischen Namen)
 - Simple security property
 - Subjekt S darf Objekt O lesen wenn $C(O) \leq C(S)$
 - *-Property
 - Subjekt S mit Lesezugriff auf O darf Objekt P schreiben, wenn $C(O) \leq C(P)$
- In anderen Worten:

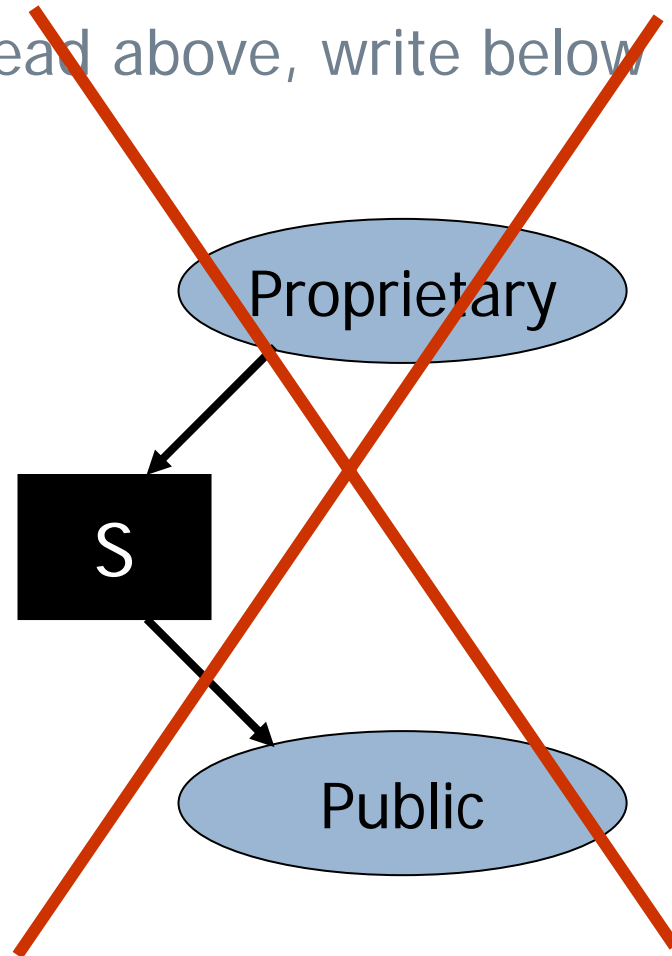
Man darf nur unter seiner Klassifikation lesen und nur über seiner Klassifikation schreiben

Bell-LaPadula Confidentiality Model (2)

Read below, write above



~~Read above, write below~~

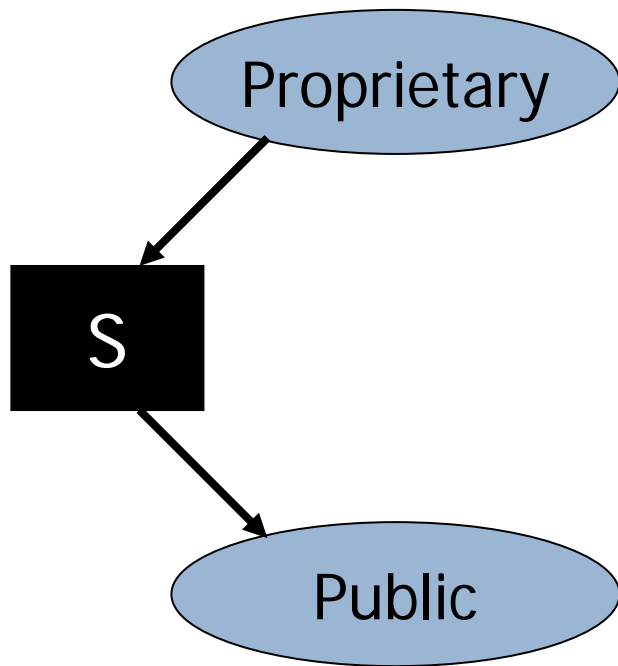


Biba Integrity Model (1)

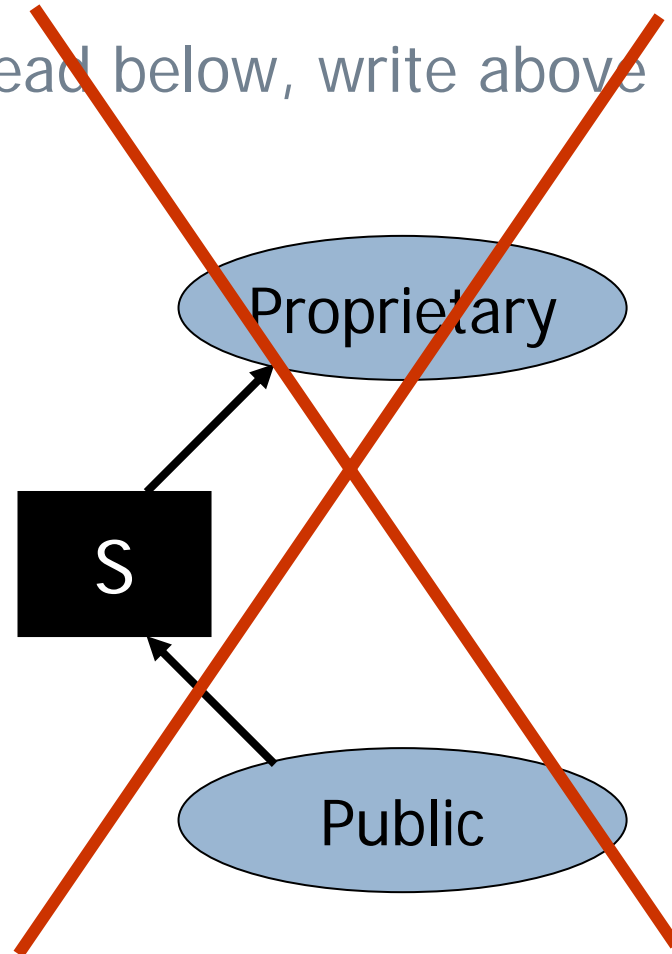
- Regeln zur Erhaltung der Datenintegrität
- Zwei Eigenschaften (mit neuer Bedeutung)
 - Simple integrity property
 - Subjekt S darf Objekt O schreiben wenn $C(S) \geq C(O)$
(S darf O modifizieren, wenn S einen höheren Rang hat)
 - *-Property
 - Subjekt S mit Lesenzugriff auf O darf Objekt P schreiben wenn $C(O) \geq C(P)$
(Daten von O nach P übertragen wenn O mindestens so vertrauenswürdig ist wie P)
- In anderen Worten:
Man darf nur unter seiner Klassifikation schreiben und nur über seiner Klassifikation lesen

Biba Integrity Model (2)

Read above, write below



Read below, write above



Problem: Modelle widersprechen sich

- Bell–LaPadula Confidentiality
 - Unten lesen, oben schreiben
- Biba Integrity
 - Oben lesen, unten schreiben
- Confidentiality und Integrity auf einmal?
 - Lesen und Schreiben nur auf der exakt selben Klassifikationsebene
- In der Praxis
 - Bell–LaPadula wird öfter genutzt als das Biba Modell
 - Beispiel: Common Criteria

Andere Konzepte

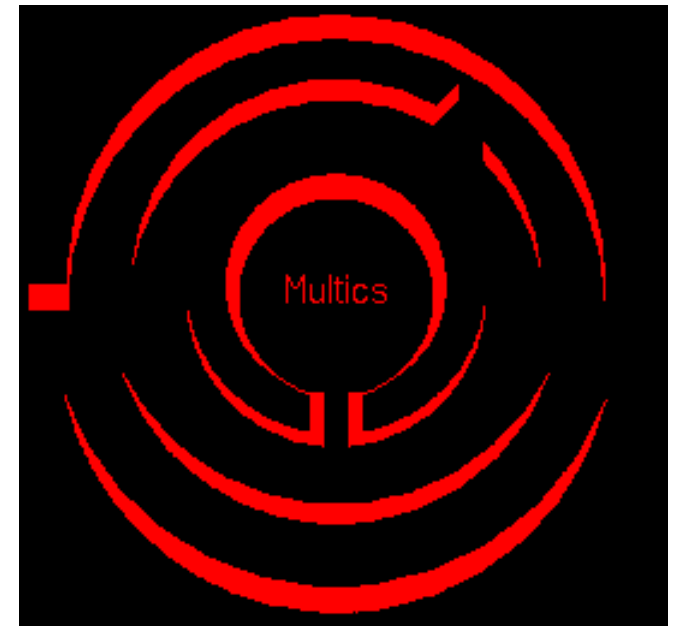
- Separation of Duty
 - Wenn der Betrag größer als \$10,000 ist, müssen zwei autorisierte Personen unterschreiben
- Chinese Wall Policy
 - Anwälte L1, L2 in Firma F sind Bankexperten
 - Wenn Bank B1 Bank B2 verklagt:
 - L1 und L2 können jeweils für B1 oder B2 arbeiten
 - Kein Anwalt kann gleichzeitig für beide Seiten arbeiten
- Diese Konzepte können nicht mit einer Access Matrix dargestellt werden

Sicherheit in Betriebssystemen

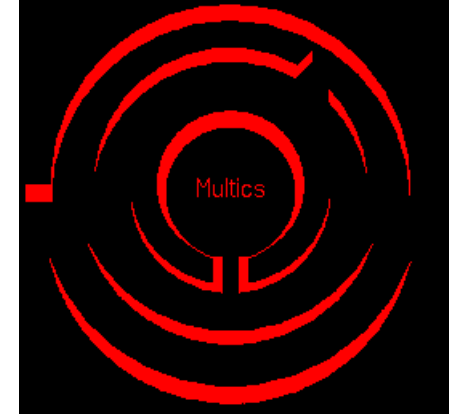
- Folgende BS werden kurz besprochen:
 - Multics
 - Amoeba
 - Unix
 - Windows

Multics

- Betriebssystem
 - Entwickelt 1964–1967
 - MIT Project MAC, Bell Labs, GE
 - Vorgänger von UNIX
 - ~100 Multics Installationen
 - Letztes System, Canadian Department of Defense, Nova Scotia, shutdown October, 2000
- Ausgeprägte Sicherheits–Mechanismen
 - Hat nachfolgende Systeme maßgeblich beeinflusst
- Mehr über Multics
 - <http://www.multicians.org>



Multics Ring Struktur



- Ring Struktur
 - Prozesse laufen in einem Ring
 - Ringe durchnummeriert 0, 1, 2, ... ; Kernel ist Ring 0
 - Graduelle Privilegien
 - Prozesse in Ring i haben Privilegien aller Ringe $j > i$
- System Aufrufe
 - User-Prozess läuft auf Ring u
 - Systemfunktion: Aufruf von Ring s nach Ring u mit $s < u$
 - Wohlspezifizierte und überwachte Eintrittspunkte von Ring u nach s

Amoeba (1)

Server port	Obj #	Rights	Check field
-------------	-------	--------	-------------

- Verteiltes System
 - Prozesse auf verschiedenen Rechnern können kommunizieren
 - Ob ein Prozess auf demselben oder einem anderen Rechner läuft spielt dabei keine Rolle
- Capability-basiertes System
 - Jedes Objekt lebt auf einem “Server”
 - Operationen werden über Nachrichten gestartet
 - Nachricht enthält Capability und Parameter
 - Server nutzt Obj# zur Objektidentifikation
 - Server prüft die Rechte anhand des Tickets
 - Check Field verhindert Veränderung des Tickets

Amoeba (2)

Server port

Obj #

Rights

Check field

- Owner Capability
 - Wenn der Server ein Objekt erzeugt, erstellt er eine Owner Capability
 - All Operationen sind erlaubt (Rights Feld)
 - Check field enthält 48-bit Zufallszahl, die auf dem Server gespeichert wird
- Derived Capability
 - Besitzer kann Bits im Right Feld löschen
 - Check Field muss neu berechnet werden
 - XOR Rights Feld mit der Zufallszahl des Check Fields
 - Anwenden einer Ein-Wege-Funktion, um das neue Check Field zu berechnen
 - Server kann Rights Feld auf seine Richtigkeit überprüfen
 - Ohne die ursprüngliche Zufallszahl kann man sich keine neuen Rechte verschaffen

UNIX

- Multi-user System
 - User darf das BS nicht korrumpieren
 - Daten verschiedener User müssen voneinander getrennt werden
- Ring Struktur
 - Nicht so ausgefeilt wie bei Multics
 - User space
 - Kernel space
- Dateisystem
 - Der wichtigste Sicherheits-Mechanismus

Unix Dateisystem Sicherheit

- Jede Datei hat Owner und Group
 - Siehe “`ls -l`” in der Shell

- Rechte

- Read, write, execute
 - Owner, Group, All
 - Darstellung als Vektor aus vier Oktalwerten



- Nur Owner und Root können Rechte ändern
 - Diese Fähigkeit kann nicht delegiert werden
- SetId bits – ... kommt gleich

Unix: Effective User ID (EUID)

- Jeder Prozess hat drei IDs
 - Real User ID (RUID)
 - Dieselbe ID wie die des Vater-Prozesses (wenn nicht zwischenzeitlich geändert)
 - Zeigt an, wer den Prozess gestartet hat
 - Effective User ID (EUID)
 - Bestimmt die Rechte des Prozesses
 - Gleich RUID, oder die EUID der Datei, welche in dem Prozess ausgeführt wird
 - Saved User ID (SUID)
 - Speichert die EUID bei Programmstart
- Ähnliches gilt wiederum für Group IDs

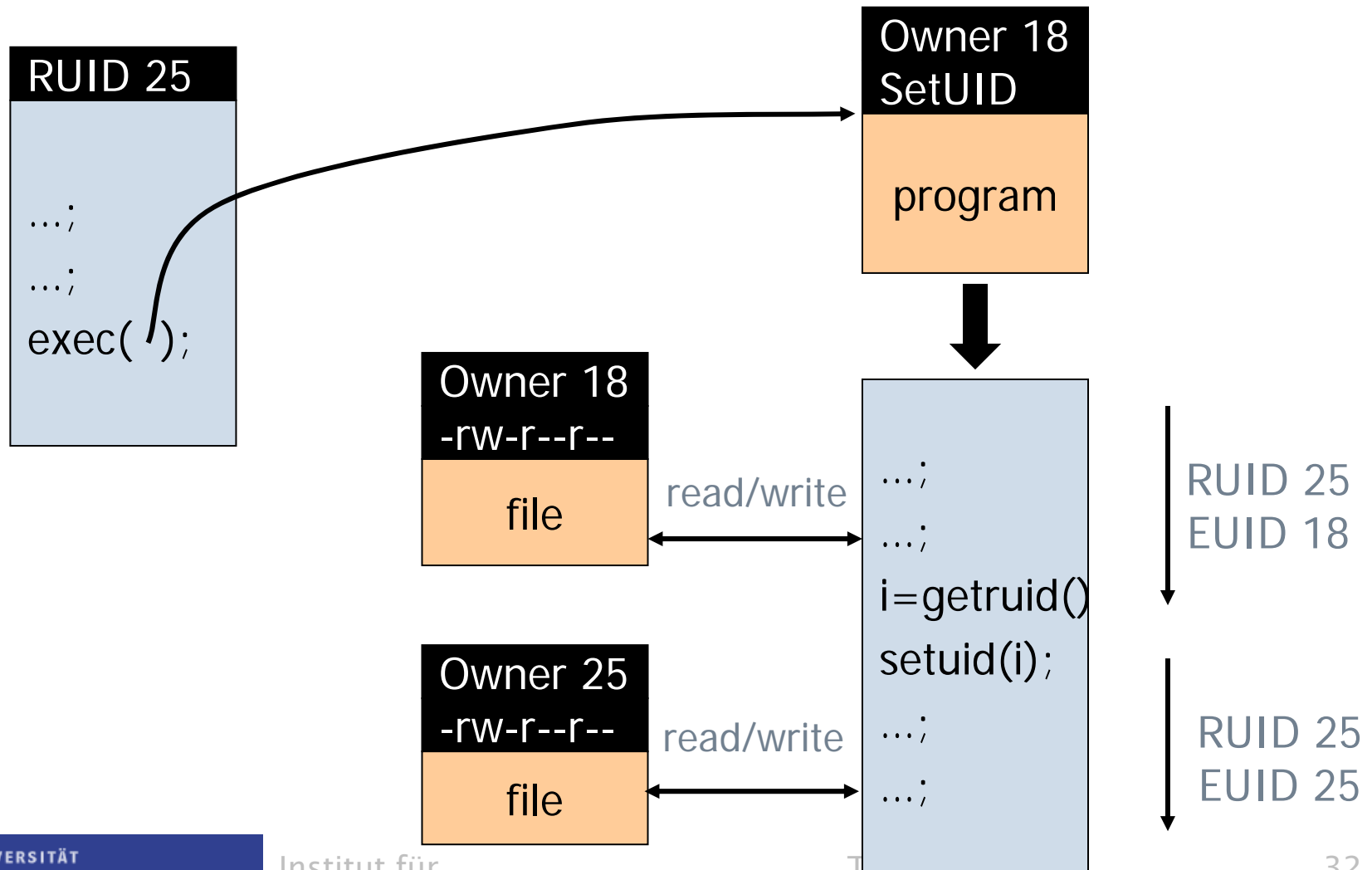
Unix: Prozess Operationen und IDs

- Root
 - ID=0 für Superuser Root; Hat alle Rechte
- Fork und Exec
 - Erbt alle drei IDs, es sei denn, das SetUID bit der Datei war gesetzt
- SetUID Systemaufruf
 - seteuid(newid) ändert die EUID
 - Root darf beliebige User ID angeben
 - Normale User dürfen zwischen Real ID und Saved ID wechseln

SetID Bits von Ausführbaren Unix Dateien (1)

- Drei SetID bits
 - SetUID – setzt die EUID des Prozesses auf die ID des Dateibesitzers
 - SetGID – setzt die EGID des Prozesses auf die GID der Datei
 - Sticky Bit
 - 0: Wenn User Schreibrechte auf dem Verzeichnis hat, dann darf er die Datei umbenennen oder löschen, selbst wenn er nicht Besitzer ist
 - 1: Nur der Dateibesitzer, Verzeichnisbesitzer oder Root können die Datei umbenennen oder löschen

SetID Bits von Ausführbaren Unix Dateien (2)



SetID Bits von Ausführbaren Unix Dateien (3)

- Vorsicht bei SetUID !
- Wer die Datei ausführen darf, tut das unter den Rechten des Dateibesitzers
- Das Programm in dieser Datei darf nicht:
 - Beliebige Aktionen ausführen
 - Sicherheitsrelevante Daten ausgeben
- SetUID nur auf Programme, deren Arbeitsweise bekannt und sicher ist
 - Nur sehr begrenzte Funktionalität
 - Source Code ging durch Code Audits

Unix Zusammenfassung

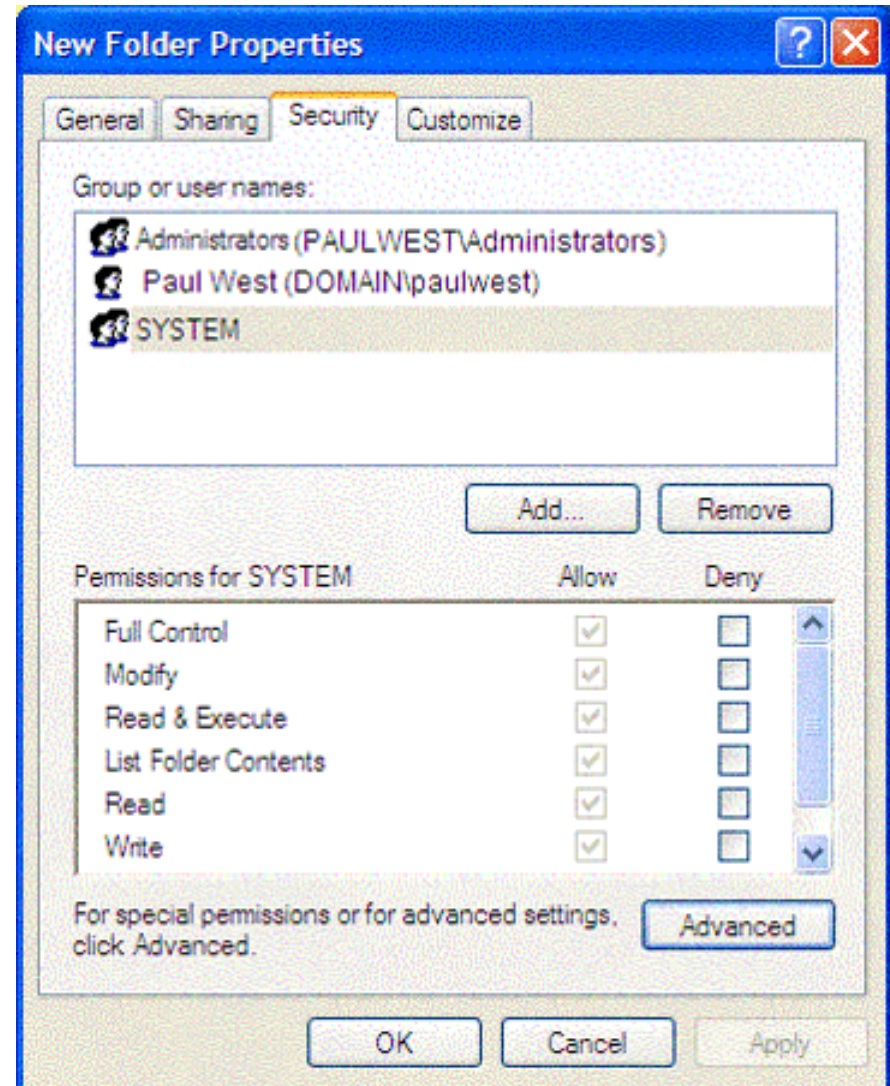
- Etablierte Technik
 - UNIX Implementierungen gelten als sehr sicher
- Positiv
 - Die Sicherheitskonzepte sind klar und einfach
- Negativ
 - Das Konzept für Gruppen ist nicht flexibel genug für moderne Geschäftsanwendungen
 - Es ist zu verlockend, als Root zu arbeiten
 - Entweder Root mit allen Rechten, oder User ohne alle Root Rechte

Windows Sicherheit

- Einiges vergleichbar mit UNIX
 - Rechte für Gruppen und User
 - Read, modify, change owner, delete
- Zusätzliche Konzepte
 - Tokens
 - Security attributes
- Etwas flexibler als Standard UNIX
 - Man kann neue Rechte definieren
 - Man kann einige aber nicht notwendigerweise alle Rechte eines Administrators haben

Windows Zugriffsschutz

- Basiert auf SID
- SID
 - Identity (ersetzt UID)
 - User, Gruppen, Computer, Domänen, etc. haben alle SIDs



Windows Permission Inheritance

- Static permission inheritance (Win NT)
 - Unterordner erben Rechte der Ordner
 - Danach können die Rechte von Ordner und Unterordner unabhängig voneinander geändert werden
 - *Replace Permissions on Subdirectories*
- Dynamic permission inheritance (Win 2000)
 - Unterordner erben Rechte der Ordner
 - Werden Rechte des Ordners geändert, dann erben auch die Unterordner diese Rechte
 - Es sei denn, der Unterordner hat explizite Rechte erhalten
 - Vererbte und explizite Rechte können im Konflikt stehen
 - Positive Rechte sind additiv
 - Negative Rechte (deny access) haben Vorrang

Windows Security Descriptor

- Objekte haben einen Windows Security Descriptor
 - Bestimmt, wer was mit dem Objekt machen darf
- Felder
 - Header
 - Descriptor Revision number, etc.
 - SID des Besitzers des Objekts
 - SID der primären Gruppe des Objekts
 - Zwei optionale Listen:
 - Discretionary Access Control List (DACL) – users, groups
 - System Access Control List (SACL) – system logs, ..

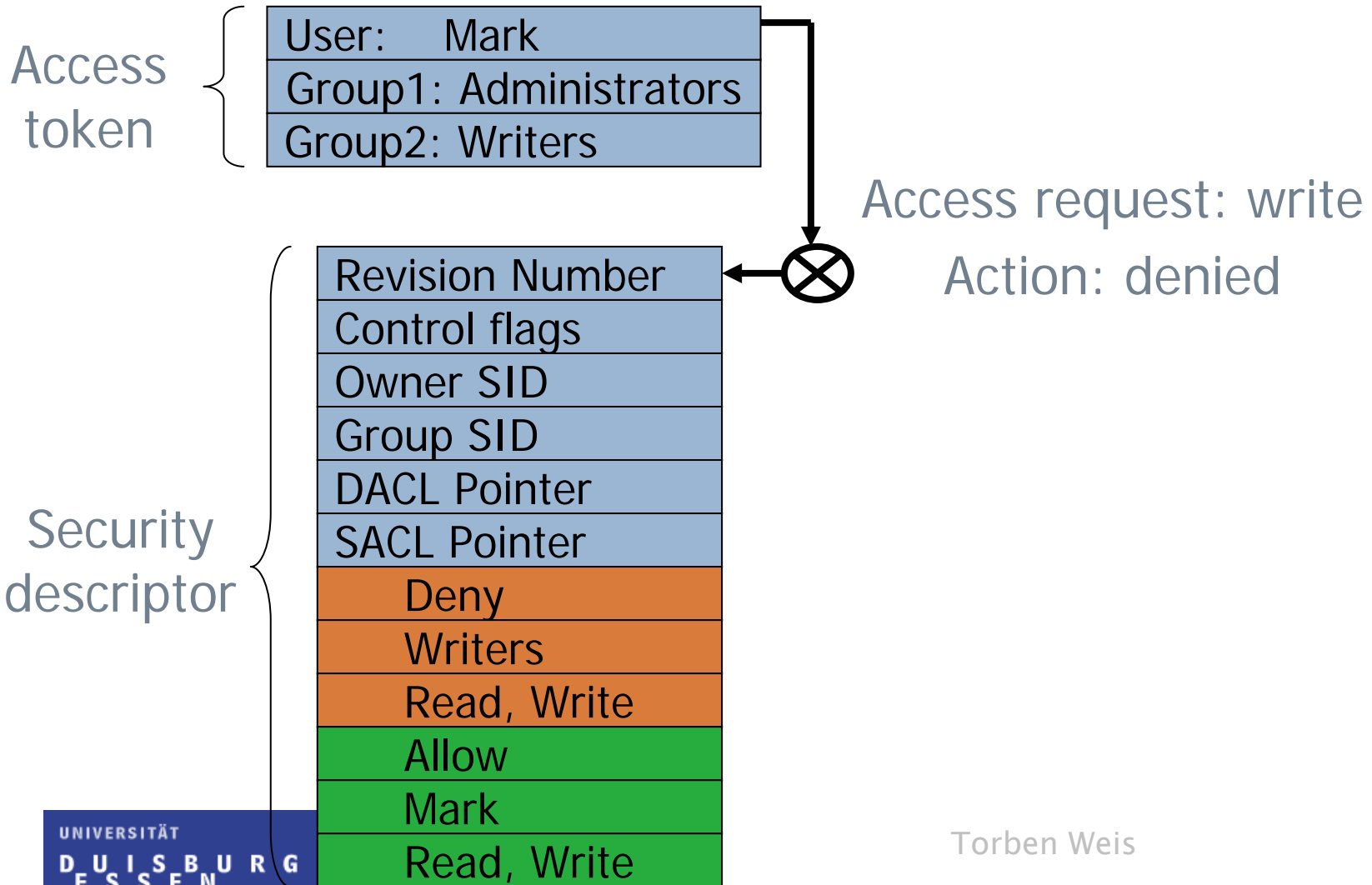
Windows Tokens

- Token beschreibt einen Sicherheitskontext
 - Identität des Users, der für die Aktion verantwortlich ist
 - Gruppenzugehörigkeit
- Prozesse benötigen ein Token, um BS Operationen ausführen zu dürfen
- Impersonation Token
 - Vergleichbar mit SetUID unter UNIX
 - Ein Prozess kann mit einem Token in den Sicherheitskontext eines Users wechseln

Windows Impersonation Tokens

- Windows Client PC kann Impersonation Token an einen Windows weiterleiten
- Client kann verschiedene Modi wählen
 - Anonymous
 - Server erfährt nicht die Identität des Clients
 - Identify
 - Server erfährt die SID des Clients und nutzt dessen Rechte
 - Impersonate
 - Server kann die Identität des Clients annehmen
 - Delegate
 - Server kann die Identität des Clients auch auf anderen Windows Servern annehmen

Beispiel: Schreibzugriff auf eine Datei



Windows: Verschlüsseltes Dateisystem

- Sehr nützlich, wenn das Laptop oder die Festplatte gestohlen werden
- Verschlüsselung
 - Symmetrischer Verschlüsselungsalgorithmus
 - Asymmetrische sind viel zu langsam
 - Symmetrischer Schlüssel wird bei Installation des BS pseudo-zufällig erzeugt
- Schlüssel Management
 - Der symmetrische Schlüssel wird mit dem öffentlichen Schlüssel eines asymmetrischen Algorithmus verschlüsselt (wrapped)
 - Der “wrapped” Schlüssel wird im Active Directory oder auf einer SmartCard gespeichert
 - Nur der Besitzer des privaten Schlüssels kann den symmetrischen Schlüssel erlangen und auf das Dateisystem zugreifen

Evaluation Assurance Levels

EAL 1: Functionally Tested

EAL 2: Structurally Tested

EAL 3: Methodically Tested and Checked

EAL 4: Methodically Designed, Tested, Reviewed

EAL 5: Semi-formally Designed and Tested

EAL 6: Semi-formally Verified Design and Tested

EAL 7: Formally Verified Design and Tested



National Information Assurance Partnership

Common Criteria Certificate



Microsoft Corporation

The IT product identified in this certificate has been evaluated at an accredited testing laboratory using the Common Methodology for IT Security Evaluation (Version 1.0) for conformance to the Common Criteria for IT Security Evaluation (Version 2.1). This certificate applies only to the specific version and release of the product in its evaluated configuration. The product's functional and assurance security specifications are contained in its security target. The evaluation has been conducted in accordance with the provisions of the NIAP Common Criteria Evaluation and Validation Scheme and the conclusions of the testing laboratory in the evaluation technical report are consistent with the evidence adduced. This certificate is not an endorsement of the IT product by any agency of the U.S. Government and no warranty of the IT product is either expressed or implied.

Product Name: Windows 2000 Professional, Server, and
Advanced Server with SP3 and Q326886 Hotfix
Evaluation Platform: Compaq Proliant ML570, ML330;
Compaq Professional Workstation AP550; Dell Optiplex
GX400; Dell PE 2500, 6450, 2550, 1550
Assurance Level: EAL4 Augmented

Name of CCTL: Science Applications International
Corporation
Validation Report Number: CCEVS-VR-02-0025
Date Issued: 25 October 2002
Protection Profile Identifier: Controlled Access Protection
Profile, Version 1.d, October 8, 1999

Director

Information Technology Laboratory
National Institute of Standards and Technology

Information Assurance
Director
National Security Agency