

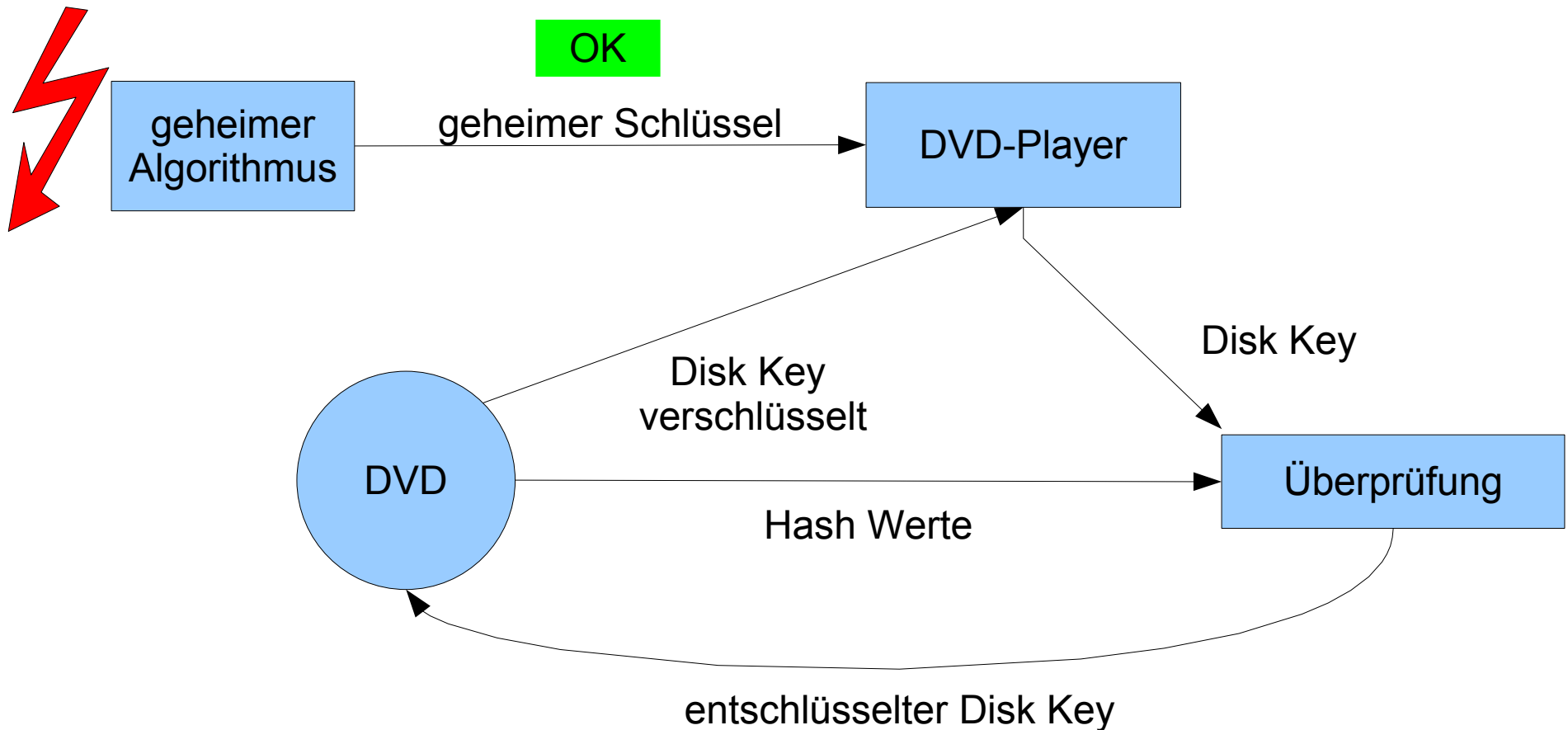
Seminar : Sicherheit und technischer Datenschutz in Informationssystemen  
Sommersemester 2006 – Uni Karlsruhe

# Entwurf sicherer Systeme

von  
Paul Mandalka

# Entwurfsfehler - Kopierschutz geknackt

- Content Scrambling System (CSS, bei DVDs)



# Prinzip - Offener Entwurf

- Offener Entwurf
  - Die Sicherheit eines Systems sollte nicht von der Geheimhaltung der Implementierung abhängen
  - Vor allem bei Verschlüsselungsalgorithmen wichtig
  - Geheimhaltung eines Schlüssels verletzt das Prinzip nicht

# Gliederung

- Entwurfsprinzipien
- Ausschlussproblem
  - Isolation
  - Versteckte Kanäle
- Entwerfen von Systemen mit Zusicherungen
  - In Anforderungsdefinition und Analysephase
  - Beim System- und Softwareentwurf
  - In der Implementierungsphase
  - Während des Einsatzes und Wartung

# Prinzip – Geringste Rechte

- Geringste Rechte
  - Nur die Rechte vergeben, die benötigt werden
  - Auch die Art der Rechte ist wichtig (z.B. Schreiben vs. Anhängen)
  - Rechte möglichst spät bekommen und möglichst früh abgeben
  - Mail-Server
    - Rechte auf : Spool-Verzeichnis, Netzwerk
    - keine Rechte : Benutzerdaten
    - Nach ablegen einer Datei, die Rechte wieder abgeben

# Prinzip - Ausfallsichere Instantiierung

- Ausfallsichere Instantiierung
  - Beim Erstellen eines Subjekt sollte dieses keine Rechte besitzen, bis es diese erhält
  - Kann ein Prozess seine Aufgabe nicht erledigen, alles rückgängig machen und sich beenden
  - Prozess soll nicht versuchen seine Rechte zu erweitern
  - Mail-Server
    - DoS-Attacken, falls Server bei vollem Spool-Verzeichnis versucht anderswo Daten zu plazieren

# Prinzip - Ökonomisch

- Ökonomie des Sicherheitsmechanismus
  - So einfach wie möglich, minimiert Fehler und Testaufwand
  - Komplexe Mechanismen machen meistens Annahmen über das System und die Umgebung, sowie Rückgabewerte oder Eingabeparameter
  - Finger-Protokoll
    - Viele Clients gehen davon aus, dass das Ergebnis wohlgeformt ist
    - Angreifer können dies durch manipulierte Antworten gezielt für DoS-Attacken einsetzen

# Prinzip - Vermittlung

- Vollständige Vermittlung
  - Alle Zugriffe auf ein Objekt müssen überprüft werden
  - Bei wiederholtem Lesen trotzdem Rechte prüfen, meistens aber nicht der Fall (Caching)
  - UNIX
    - Beim Zugriff auf eine Datei, erhält das Subjekt einen Datei-Descriptor
    - Der Datei-Descriptor wird dann für weitere Zugriffe verwendet, was jedoch nicht mehr überprüft wird

# Prinzip - Rechteaufteilung

- Aufteilung von Rechten
  - Das System sollte Rechte nicht anhand **einer** Bedingung einräumen
  - Berkeley basierte UNIX Systeme
    - Benutzer kann erst „root“ Rechte erhalten falls er
      - Das root-Passwort kennt
      - Benutzer ist in der Administratoren Benutzergruppe (GID 0)

# Prinzip - Akzeptanz

- Psychologische Akzeptanz
  - Zugriff auf das System sollte für den Benutzer nicht viel komplizierter werden
    - Speichern von Verbindungsdaten (public key - *ssh*)
  - Sicherheitseinstellungen sollten einfach zu bedienen sein
    - Kann zu unbeabsichtigten falschen Einstellungen führen
  - Fehlermeldungen sollten sprechend sein, jedoch nicht zu viel preisgeben
    - Beim falschen Anmelden nur „Login fehlgeschlagen“ ausgegeben werden, anstatt z.B. „Passwort falsch“

# Prinzip - Wenige gemeinsame Funktionen

- Wenige gemeinsamen Funktionen
  - Mechanismen zum Zugriff auf Ressourcen sollten nicht geteilt werden
  - Teilen von Ressourcen impliziert einen Kanal, über den Informationen ausgetauscht werden können
  - Dateisystem
    - Mit Hilfe von Datei-Namen oder Attributen kann eine Kommunikation ermöglicht werden

# Gliederung

- ✓ Entwurfsprinzipien
- Ausschlussproblem
  - Isolation
  - Versteckte Kanäle
- Entwerfen von Systemen mit Zusicherungen
  - In Anforderungsdefinition und Analysephase
  - Beim System- und Softwareentwurf
  - In der Implementierungsphase
  - Während des Einsatzes und Wartung

# Definitionen

- Das Ausschlussproblem
  - Vertrauliche Informationen schützen
  - Nur berechnigte Prozesse dürfen Daten erhalten und weitergeben
- Isolation
  - Virtuelle Maschinen
    - Kommunikation wird kontrolliert
  - Sandkasten
    - Erweitern der Bibliotheken oder des Kernel um Sicherheitsfunktionen

# Versteckte Kanäle

- Versteckte Kanäle
  - Kommunikationspfad, der nicht zu Kommunikation entwickelt wurde
- Versteckte Speicher Kanäle
  - benutzt Attribute von geteilten Ressourcen
- Versteckte Kanäle durch zeitliche Abstimmung
  - Benutzt temporale oder angeordnete Beziehungen
  - Durch Echtzeituhr, Zeitgebern oder Ereignisabfolgen

# Versteckte Kanäle - Beispiel

- Zeit kann als Informationsträger dienen
  - Analyse der Laufzeit eines Programms
  - Beispiel, berechnet  $x = a^z \bmod n$

```
x := 1; atmp := a;
for i := 0 to k-1 do begin
  if zi = 1 then
    x := (x * atmp) mod n;
    atmp := (atmp * atmp) mod n;
end;
result := x;
```

Bei  $z_i = 1$  werde  
2 Multiplikationen berechnet

- Laufzeitunterschiede ermöglichen Rückschlüsse

# Versteckte Kanäle finden

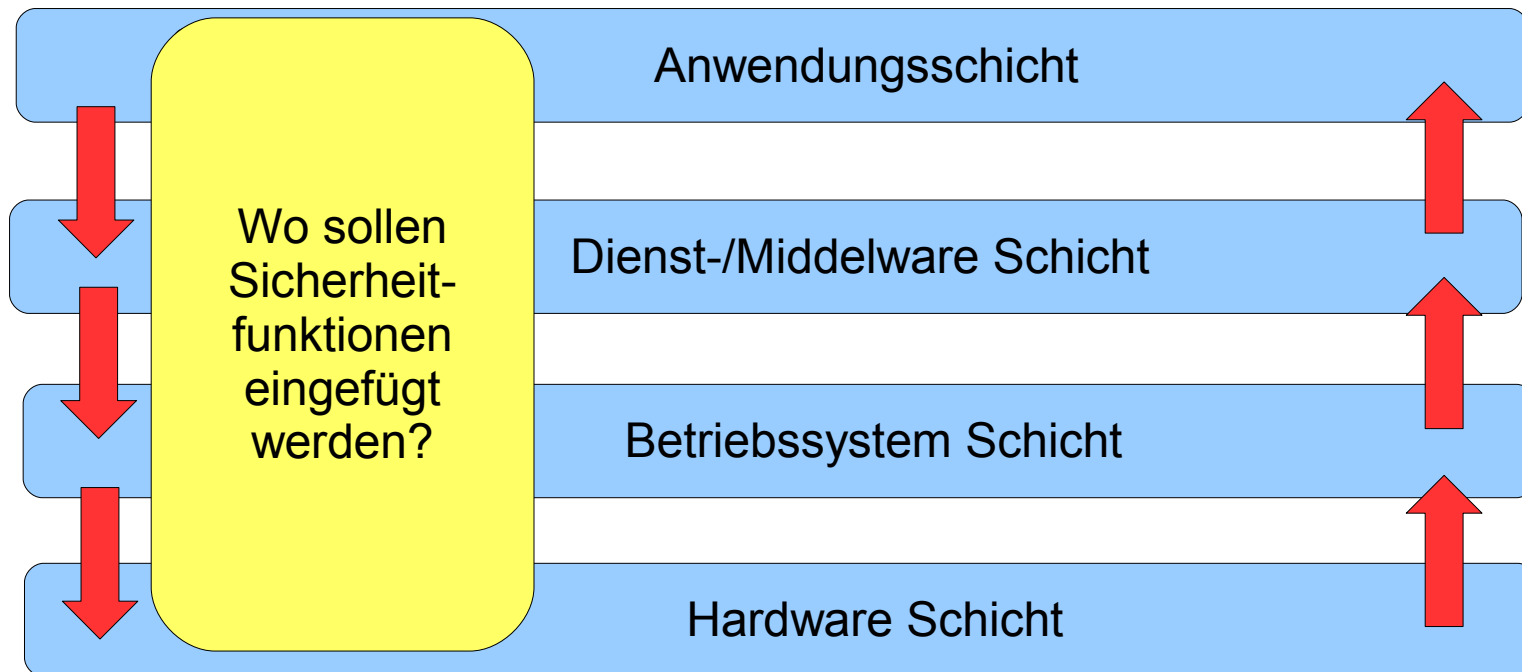
- Shared Resource Matrix Methode (SRM)
    - Matrix mit Attributen als Zeilen und Operationen als Spalten
- |                | read_file | write_file | delete_file | create_file |
|----------------|-----------|------------|-------------|-------------|
| file existence | R         | R          | <b>R, M</b> | <b>R, M</b> |
| file owner     |           |            | R           | M           |
| file label     | R         | R          | R           | M           |
| file size      | R         | M          | M           | M           |
- Attribute die sowohl geändert (M) als auch gelesen werden können (R) sind Problemstellen
  - Kriterien entscheiden, ob es einen versteckten Kanal gibt

# Gliederung

- ✓ Entwurfsprinzipien
- ✓ Ausschlussproblem
  - ✓ Isolation
  - ✓ Versteckte Kanäle
- Entwerfen von Systemen mit Zusicherungen
  - In Anforderungsdefinition und Analysephase
  - Beim System- und Softwareentwurf
  - In der Implementierungsphase
  - Während des Einsatzes und Wartung

# Anforderungsdefinition und Analysephase

- Wo sollen Sicherheitsfunktionen eingefügt werden



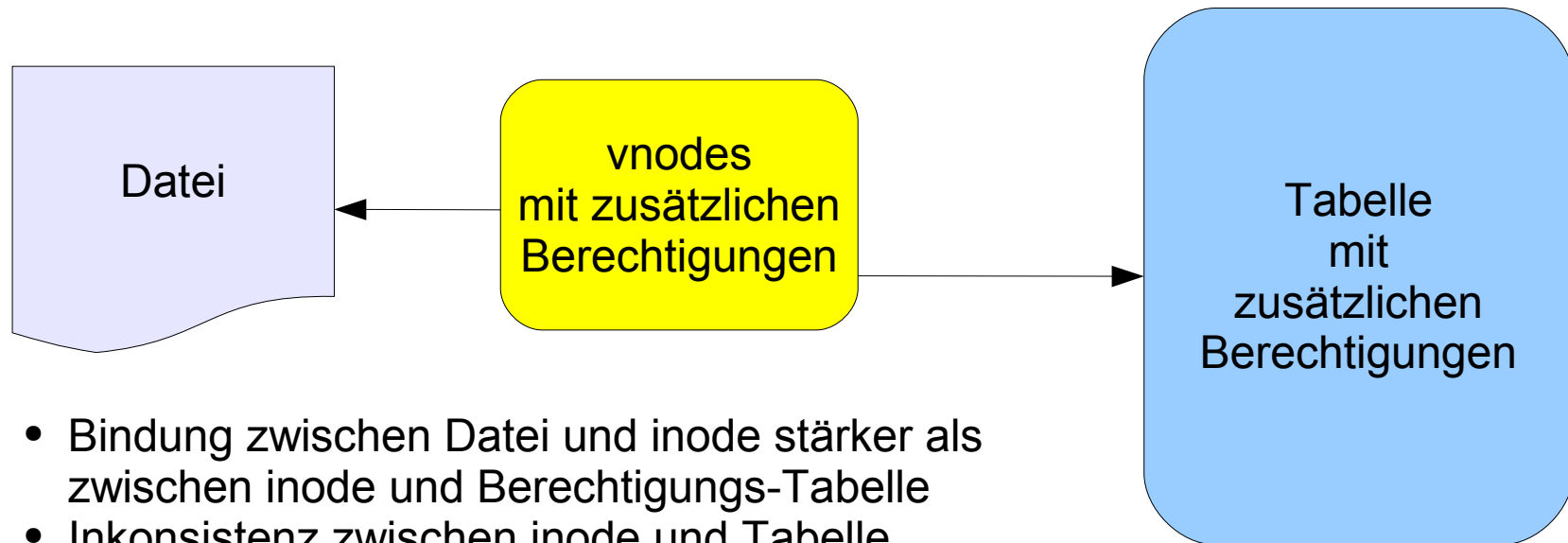
# Anforderungsdefinition und Analysephase (2)

- Wann sollen Sicherheitsfunktionen eingefügt werden
  - Während des Entwurfs
    - evtl. mehr Planungsaufwand
    - sicherere Systeme
  - Im Nachhinein
    - Entwurfsentscheidungen können die Sicherheit beeinflussen
    - evtl. schneller, wenn System besteht
    - Kann aber zu unsichereren Systemen führen

# Beispiel – Sicherheit im Nachhinein

- AT&T wollte UNIX System V um Sicherheitsfunktionen erweitern

1. Ansatz : Erweiterung des Systems



- Bindung zwischen Datei und inode stärker als zwischen inode und Berechtigungs-Tabelle
- Inkonsistenz zwischen inode und Tabelle
- Fehler in Tabelle beeinflussen ganzes System

2. Ansatz : inode Restrukturierung

# Entwurf von Richtlinien (Policy)

- Richtlinien erfüllen Spezifikationen und Anforderungen
- Vorgehen
  - (1) Aus ähnlichen Projekten herausnehmen
  - (2) Erstellen eine neuen Richtlinie anhand existierender Richtlinien und einer Bedrohungsanalyse
  - (3) Abbildung des Systems auf ein existierendes Modell

# Überprüfung

- Richtlinien muss auf Vollständigkeit und Konsistenz überprüft werden
- **Information Technology Security Evaluation Criteria – ITSEC**
  - informale Beschreibung von Bedrohungen (T), Anforderungen (IA) und Annahmen (A)
  - Tabellarische Zuordnung

Threat	Security Target Reference
T1	IA1, IA2, A1, A2, A3, A4
  - <http://www.bsi.de/zertifiz/itkrit/itsec.htm>

# Beim System- und Softwareentwurf

- Modularisierung und Schichtenarchitektur erhöhen die Sicherheit
  - Kleinere Komponente sind übersichtlicher, verständlicher und einfacher zu warten
  - Ermöglicht besseres Data-Hiding
  - Fördert das Entwurfsprinzip „Geringste Rechte“

# Entwurfsdokumentation

- Elemente der Dokumentation

- (1) Sicherheitsfunktionen

- Beschreibung einzelner Funktionen
    - Zusammenarbeit der einzelnen Funktionen
    - Abbildung auf die Sicherheitsanforderungen

- (2) Externe Schnittstellen

- Komponentenübersicht
    - Datenbeschreibung
    - Schnittstellenbeschreibung

# Entwurfsdokumentation (2)

## (3) Interne Entwurfsbeschreibung

- Übersicht über Eltern-Komponente
- Detaillierte Beschreibung der Komponente
- Sicherheitsrelevanz der Komponente

## (4) Interne Entwurfsspezifikation

- Informeller als die anderen
- Meistens nur für Low-Level-Dokumentation

# Beispiel für Dokumentation

- Externe Schnittstellen

- Interface Name

- Input Parameters

- handle – a valid handle returned from a previous call to `open_log`
- event – the buffer of event data with event records in logevent format

- Exceptions

- Caller does not have permission to add to EVENT file

- Effects

- Event is added to EVENT log

- Output Parameters

- status : `status_ok`, `no_memory`, `permission_denied`

- Note :

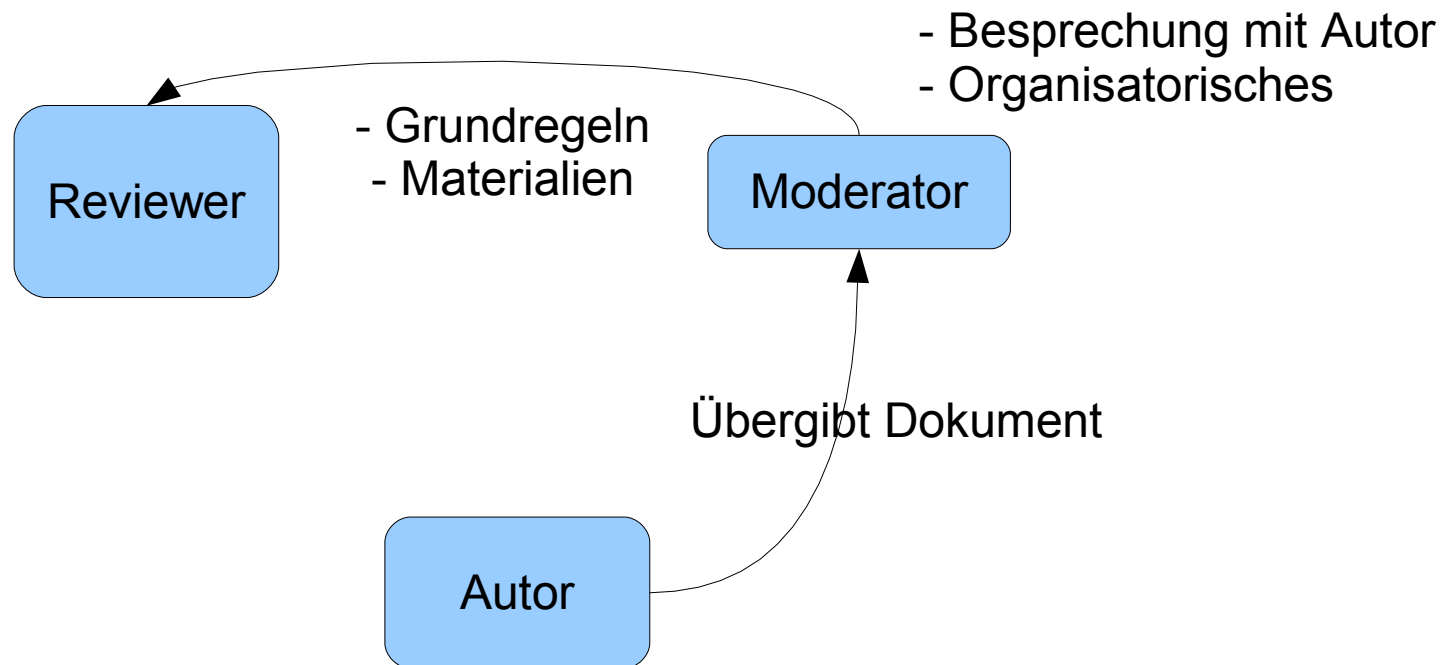
- `add_logevent` is a user-visible interface

```
error_t add_logevent (  
    handle_t handle,  
    data_t event  
);
```

# Überprüfung des Dokumentes

- Reviews

- Da Dokumente meistens informell
- Vorgehen - Vorbereitung



# Überprüfung des Dokumentes (2)

- Reviews
  - Vorgehen – Technischer Review
    - Analyse des Dokuments anhand der Grundregeln und Richtlinien
    - Kommentare der Reviewer
  - Das Review Meeting
    - präzise Wiedergabe der Kommentare
    - Konstruktiv und nicht erniedrigend oder persönlich
    - Sammlung an Kommentaren, keine Lösung

# Überprüfung des Dokumentes (3)

- Konfliktlösung
  - Abstimmung über Kommentare
  - Reviewer sollen Chance zur Diskussion erhalten
  - Autor versucht das Dokument anhand der Kommentare zu ändern
  - Autor und Reviewer überprüfen, ob die Änderungen erfolgreich waren
- Ende des Reviews
  - Nachdem alle Konflikte gelöst sind

# In der Implementierungsphase

- Programmiersprache kann Sicherheit beeinflussen
  - Einsatz von low-level Sprachen manchmal nötig
  - Coding Standards
- Konfigurationsmanagement
  - Versionkontrolle (z.B. CVS, SVN)

# Überprüfung

- Sicherheitsüberprüfung
  - Funktionale Tests (black box)
    - Überprüfung wie gut Spezifikation eingehalten wird
  - Struktur Tests (white box)
    - Code-Analyse zum Entwurf von Test Szenarien

# Während des Einsatzes und der Wartung

- Hot-Fixes
  - möglichst schnelle Lösung
  - Bei Sicherheitsmängeln oder wenn Funktionalität gefährdet ist
- Maintenance Releases
  - weniger kritische Bugs oder Verbesserungen von hot-fixes
- Sicherheit des Produkts muss überprüft werden

# Zusammenfassung

- Entwurfsprinzipien helfen einige Fehler zu vermeiden
- Auch unscheinbare Funktionen können Probleme verursachen (versteckte Kanäle)
- Übersichtliches Design und gute Dokumentation sind wichtig
- Überprüfungen von Entwicklungsphasen sind wichtig

# Fragen ?

