»Alternative« Methoden zur Erhöhung der Sicherheitstransparenz [im Open-Source-Bereich]

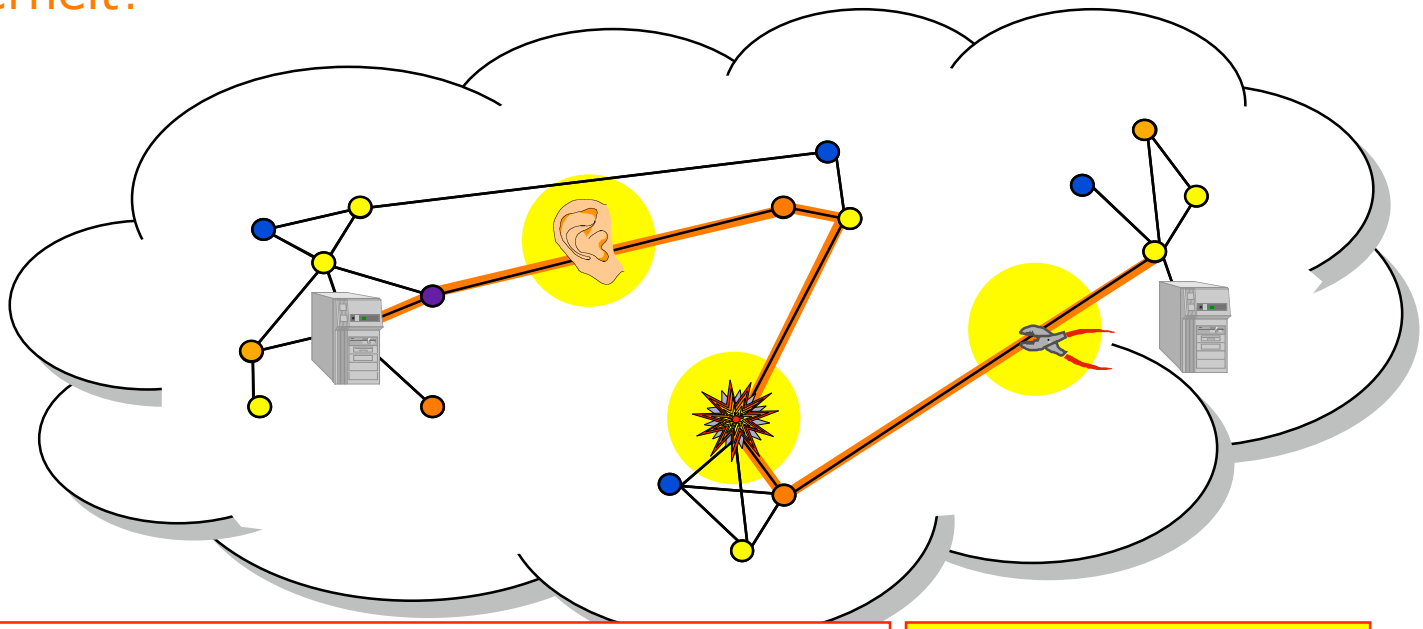
Hannes Federrath

http://www.inf.fu-berlin.de/~feder/

- Einführung
- OS und Evaluation von Software
- Formale Spezifikation und Verifikation sicherheitskritischer Funktionen
- Sicherheitsschnittstellen
- Schlussbemerkungen

Problemstellung

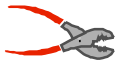
⌘ Was ist Sicherheit?



Bedrohungen

 unbefugter Informationsgewinn

 unbefugte Modifikation

 unbefugte Beeinträchtigung der Funktionalität

Schutz der

Vertraulichkeit

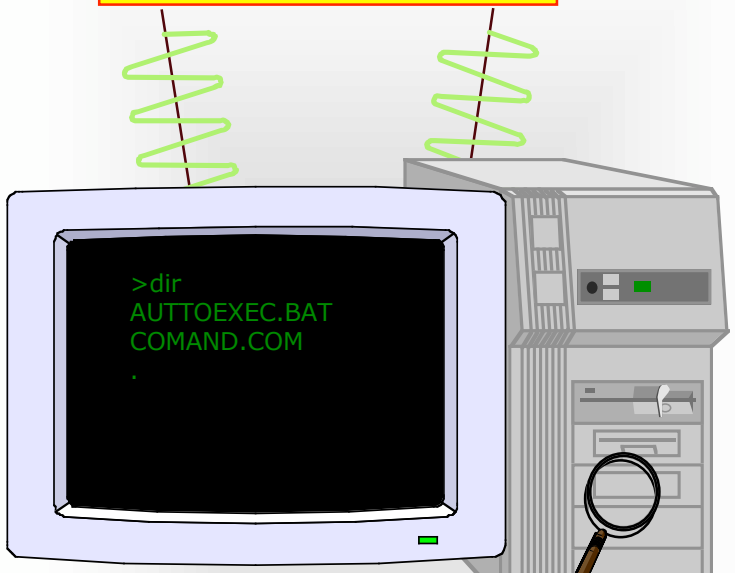
Integrität

Verfügbarkeit

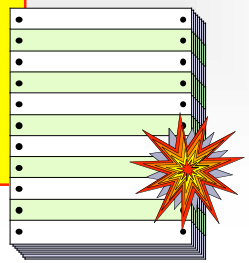
Angriffspunkte

Rechner

Abstrahlung

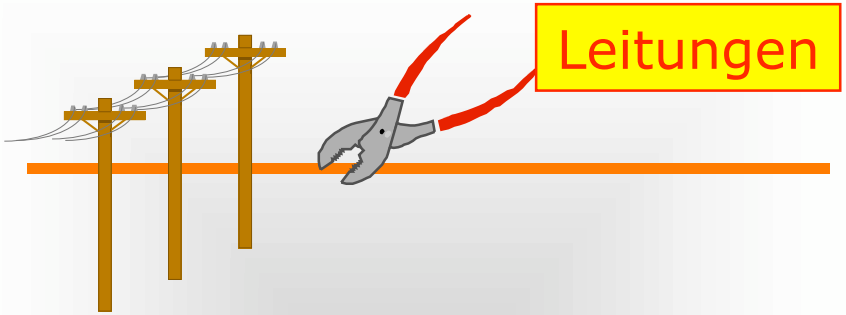


innen (Trojanische Pferde)

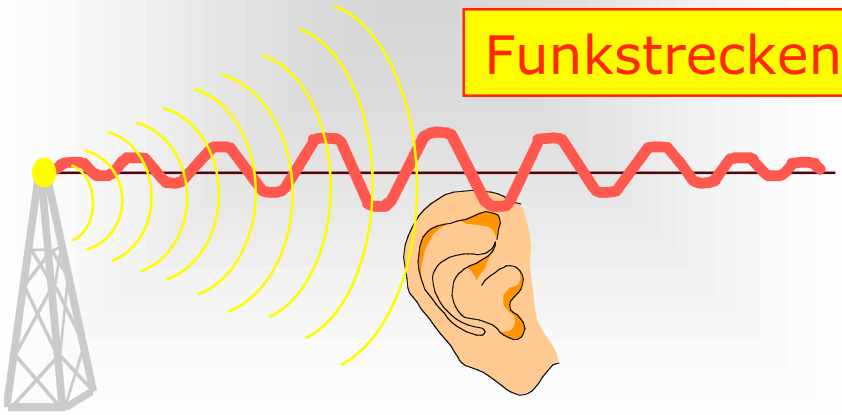


Übertragungswege

Leitungen



Funkstrecken



Sicherheit: Abgrenzung von Security & Safety

SECURITY
Schutz gegen beabsichtigte Angriffe

Vertraulichkeit
- Abhörsicherheit
- Sicherheit gegen unbefugten Gerätezugriff
- Anonymität
- Unbeobachtbarkeit

Integrität
- Übertragungsintegrität
- Zurechenbarkeit
- Abrechnungsintegrität

Verfügbarkeit
- Ermöglichen von Kommunikation

SAFETY
Schutz vor unbeabsichtigten Ereignissen

Fehlertoleranz

Verfügbarkeit
- Funktionssicherheit
- Technische Sicherheit
- Schutz vor Überspannung, Überschwemmung, Temperaturschwankungen
- Schutz vor Spannungsausfall

Sonstige Schutzziele
- Maßnahmen gegen hohe Gesundheitsbelastung
- …

OS und Evaluation von Software

⌘ Kriterien und Zertifikate

- Evaluation einer Software nach vorgegebenen Kriterien
  IT-Sicherheits-Evaluationskriterien, Beispiele:
  - Common Criteria (CC)
  - Information Technology Security Evaluation Criteria (ITSEC)
  - Trusted Computer Systems Evaluation Criteria (TCSEC)

- Neue Idee: (heute so noch nirgendwo zu finden)
  - Generelle Offenlegung des Quellcodes (für Jedermann, um ihn zu prüfen, nicht unbedingt gleichzusetzen mit der freien Weiterverwendbarkeit) als neue Anforderung?

- Sinn der Evaluationskriterien:
  - geordnete und nachvollziehbare Durchführung von Evaluationen durch Akkreditierte Evaluationsstellen
  - keine Notwendigkeit der Offenlegung für »Jedermann«

OS und Evaluation von Software

Bedeutungen der Assurance Levels der CC

EAL 0	Inadequate Assurance
EAL 1	Functionally Tested
EAL 2	Structurally Tested
EAL 3	Methodically Tested and Checked
EAL 4	Methodically Designed, Tested and Reviewed
EAL 5	Semiformally Designed and Tested
EAL 6	Semiformally Verified Design and Tested
EAL 7	Formally Verified Design and Tested

Äquivalenzen der Assurance Levels

Evaluation Levels der ITSEC

- E0 Inadequate Assurance
- E1 Security target and informal architectural design
- E2 Informal detailed design, and test documentation
- E3 Source code or hardware drawings
- E4 Formal model of security and semi-formal specification of security enforcing functions, architecture and detailed design
- E5 Architectural design explains the inter-relationship between security enforcing components
- E6 Formal description of architecture and security enforcing functions

Äquivalenzen der Assurance Levels

⌘ Kriterien und Zertifikate

- Immerhin: Zugriff auf Source Code (für den Evaluator) für bestimmte Assurance Families und Components
  - Beispiele: ISO IS 15408 (Common Criteria) Teil 3
    - Assurance Class ADV: Development
      - » Assurance Family INT: TSF internals
      - » Assurance Family IMP: Implementation representation
      - » Assurance Family LLD: Low-level design

- Fazit:
  - CC: mittlere bis höhere Evaluationslevel (ab EAL4) sind ohne Offenlegung des Quellcodes nicht erreichbar
  - ITSEC: Offenlegung des Source Codes ab E3 gefordert

OS und Evaluation von Software

- Kriterien und Zertifikate
  - Um Missverständnissen vorzubeugen: Keine Offenlegung für Jedermann, sondern nur für den Evaluator
  - Problem: Unterschiedliche *Ideologien* der beiden Lager:
    - Open Source Community eher anarchisch organisiert
    - Evaluationskriterien-Community ist eher militärisch-bürokratisch orientiert
  - Durchführungsfragen:
    - Wer zahlt für den Prozess?
    - Was kommt am Ende heraus (OS-Zertifikat)?
    - Wie wird der Reviewprozess organisiert?
      - Kommunikation zwischen Evaluator und Entwickler ist wesentlicher Bestandteil des Evaluationsprozesses, der dann n-fach vorkommen würde.
- Frage: gibt es einen Markt für einen organisatorisch und inhaltlich sauberen OpenSource-Review-Prozess, der am Ende etwas Nützliches zustandebringt?

OS und Evaluation von Software

- Gütesiegel
  - »weichere« Form der Zertifizierung
  - aktuell im Bereich Datenschutz

Hersteller einer Software kann sich Datenschutzfreundlichkeit eines Produkts bescheinigen lassen

  - Produkteigenschaften:
    - Erforderlichkeit
    - Datenvermeidung
    - Datensparsamkeit
    - IT-Sicherheit

OS und Evaluation von Software

- Gütesiegel
  - Datenschutz-Gütesiegel-Initiativen
    - Unabhängiges Landeszentrum für Datenschutz Schleswig-Holstein (http://www.datenschutzzentrum.de/guetesiegel/)
    - quid! Gütesiegel (http://www.quid.de/)
  - Wer prüft?
    - unabhängige akkreditierte Sachverständige
    - Prüfstellen, z.B. Prüfstelle für Datenschutz (PfDS) der TÜViT (http://www.tuvit.de/)
  - Besonderes Problem:
    - Im Datenschutz sind Vertraulichkeitseigenschaften besonders wichtig!
    - Wie kann man sicher sein, dass eine Software keinen verdeckten Kanal besitzt?

Aufbau eines Gutachtens

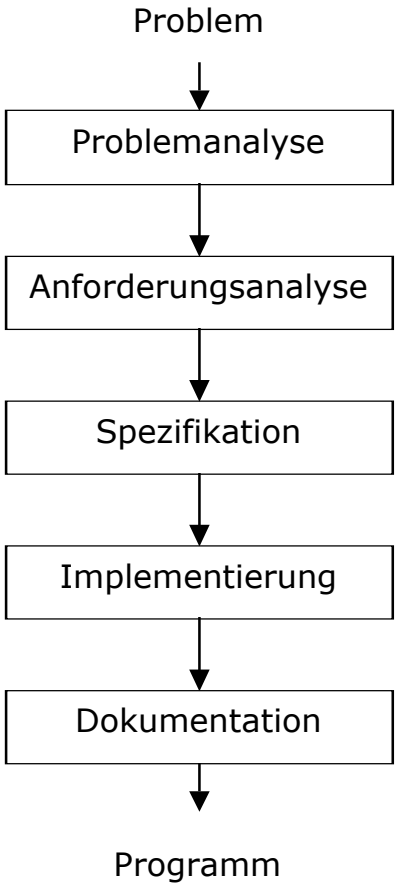
1. Zeitpunkt der Prüfung,
2. detaillierte Bezeichnung des IT-Produktes,
3. Zweck und Einsatzbereich,
4. besondere Eigenschaften des IT-Produktes, insbesondere zur
   - Datenvermeidung und Datensparsamkeit,
   - Datensicherheit und Revisionsfähigkeit der Datenverarbeitung,
   - Gewährleistung der Rechte der Betroffenen,
5. Bewertung der besonderen Eigenschaften,
6. Zusammenfassung der Prüfergebnisse

Die Einhaltung technischer und rechtlicher Anforderungen wird bestätigt.

Die Überprüfbarkeit von Vertraulichkeitseigenschaften erfordert fast zwangsläufig die Offenlegung des Quellcodes gegenüber dem Prüfer.

- Von der Sicherheits-spezifikation zum fertigen (zuverlässigen) Produkt (in der Theorie)
  - Es gibt keinen Algorithmus zum Schreiben eines Algorithmus
  - Gibt es einen Algorithmus zum Beweisen eines Algorithmus?
    - automatisierte Beweisverfahren?

Phasen des Softwareentwicklungsprozesses



Dokumentation beschreibt:

Welche generellen Leistungen soll das System erbringen?

Was soll das System im einzelnen leisten?

Wie funktioniert das System?

⌘ Zuverlässige SW, Ansätze zur Verifikation

⌦ Integrität und Verfügbarkeit:

Zeigen von Korrektheitseigenschaften

- partielle Korrektheit:
  Wenn der Algorithmus ein Ergebnis liefert, dann ist es richtig
  … ist eine reine Integritätseigenschaft

- totale Korrektheit:
  Der Algorithmus liefert ein richtiges Ergebnis
  … ist Integrität und Verfügbarkeit zusammen
  … unter Einhaltung von Zeitschranken!

- Zuverlässige SW, Ansätze zur Verifikation
  - Vertraulichkeit:
    - Kommunikation wird modelliert als System nebenläufiger Prozesse: CSP(Communication Sequential Processes)
    - Prozesse: Sender, Empfänger, Netzbetreiber, Kommunikationskanal, Angreifer
    - Informationsflussanalyse:
      - Vertraulichkeit beweist man, indem man zu zeigen versucht, das keine Kommunikation zwischen Angreiferprozess und den berechtigten Systemteilen erfolgt
- Bootstrapping-Problem:
  - Woher weiß ich, dass das Beweissystem korrekt arbeitet?

Sicherheitsschnittstellen

- Modularisierung und Komponentenbildung
  - Anstelle alles selber zu programmieren, auf zuverlässige Komponenten zurückgreifen

- Kryptobibliotheken
  - RSAREF
  - Cryptix
  - CryptoManager++
  - SecuDE

  - neuer Trend:
    Modularisierung der Funktion, nicht mehr nur des Algorithmus:
    - Auffassen eines kryptographischen Algorithmus als abstrakter Datentyp
    - wenn Alg. 1 unsicher ist, dann »umschalten« auf Alg. 2

Sicherheitsschnittstellen

- Security-APIs
  - Java-Security-API
  - SSL
  - GSS-API
  - MS-Crypto-API
  - Chipkarten

- Prüfen der zugesicherten Eigenschaften einer Komponente
  - Digitale Signatur des Codes
    - Schutz der Unverfälschtheit und Echtheit
    - jedoch keine Korrektheit der Funktionalität
  - »Normaler« Spezifikations- und Verifikationsprozess wäre möglich
  - Alternative: Proof Carrying Code: Programmcode, der einen formalen Beweis seiner Eigenschaften mit sich führt
    - Proofchecker lokal oder bei vertrauenswürdiger dritter Stelle

Zusammenfassung

- Sicherheit
  - Vertraulichkeit, Integrität, Verfügbarkeit
- Open Source und Evaluationskriterien
  - Die Communities könnten sich aufeinander zubewegen…
- Was kann man beweisen, was nicht?
  - Integritätseigenschaften: sehr gut und sehr zuverlässig
  - Verfügbarkeitseigenschaften: ebenso
  - Vertraulichkeitseigenschaften: bisher kaum oder gar nicht
- Sicherheitsschnittstellen und fertige Komponenten verwenden
  - im Sicherheitsbereich immer noch besser, als typische Anfängerfehler bei der Implementierung von Sicherheitsfunktionen zu machen

Open Source ist zweifellos eine vertrauensbildende Maßnahme.