Separation Kernels

Grundlagen und Charakteristika

Markus Burghart
markus.burghart@fernuni-hagen.de

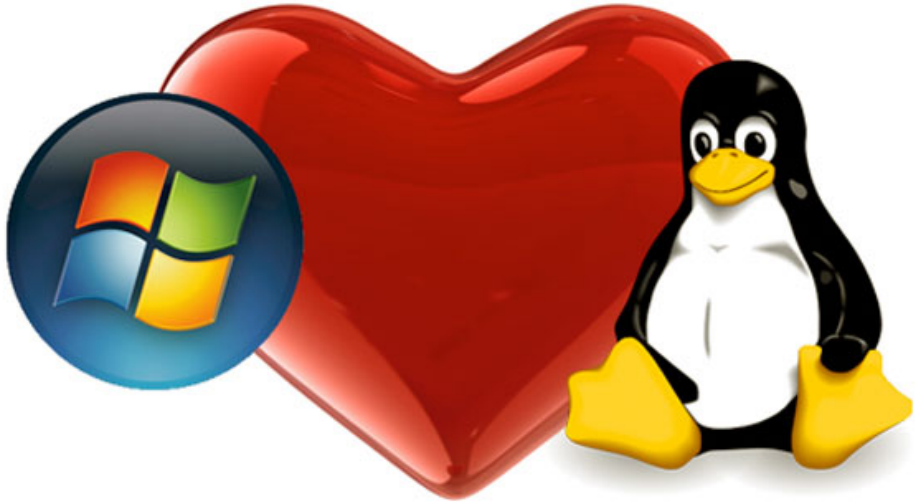
Daniel Maslowski
info@orangecms.org

September 6, 2013

Übersicht

1. **Kernel (Betriebssystem)**
   - Aufgaben des Kernels
2. **Systemsicherheit**
   - Zugriffskontrollen
   - Security Kernel
3. **Common Criteria (CC)**
   - Evaluation Assurance Levels (EAL)
4. **Separation Kernel (SK)**
   - Charakteristika
   - Separation Kernel Protection Profile (SKPP)
   - Verifizierung
   - Anwendungsbereiche

Kernel: Grundfunktionen



- Kernkomponente eines Betriebssystems
- nimmt wenig Speicherplatz ein
- Schnittstelle für die Kommunikation mit Hardware
- Systemaufrufe für Benutzer und Prozesse

Kernel: Schema

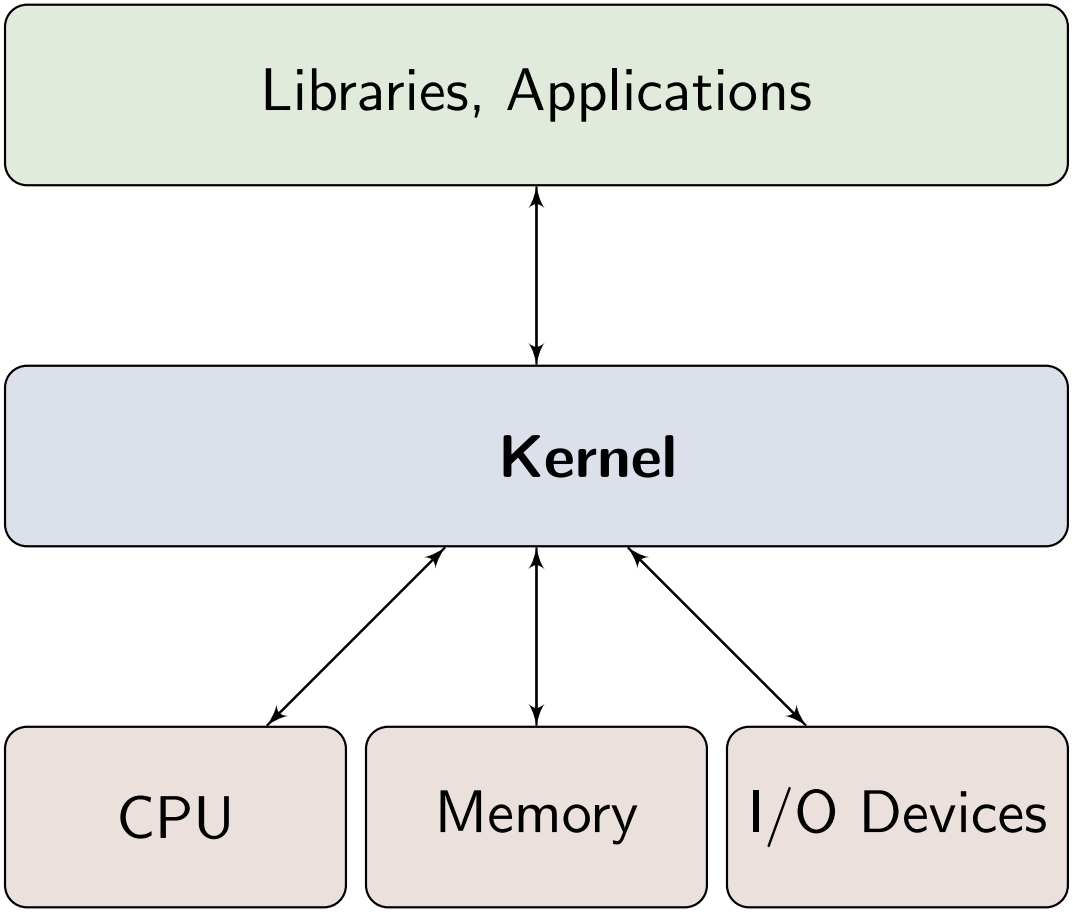


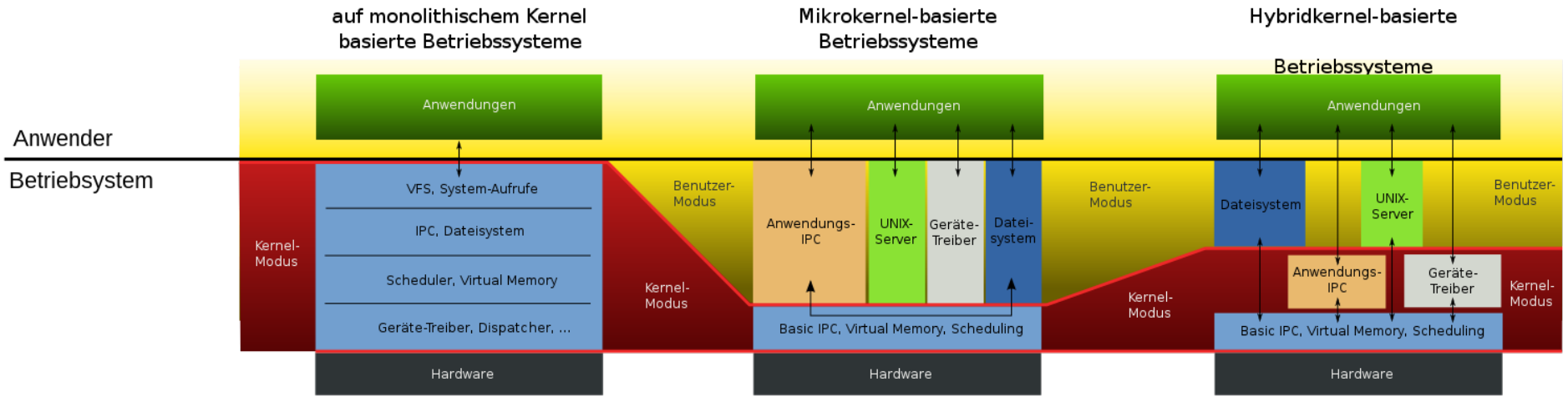
Figure : Schema eines Computersystems

Kernel: Details

- Virtualisierung der Ressourcen
- Strukturierung der Ressourcen
- Verteilung der Ressourcen
- Verwaltung von Speicher, Geräten und Prozessen
  - Gerätetreiber
  - Dateisysteme
- Multitasking, Multiuser-Anmeldung
- Überwachung von Zugriffsrechten
- Auflösung von Zugriffskonflikten

Kernel: Typen von Kernels

- Mikrokernel
  - Grundfunktionen: Speicher- und Prozessverwaltung, Basis-IPC
  - sehr klein und robust
- Monolithischer Kernel
  - weitere Funktionen: Treiber, Module, Dateisysteme
  - größer, erhält Stabilität durch Modularisierung
- Hybridkernel
  - Mischlösung zwischen monolithischem und Mikrokernel



Zugriffskontrollen I

zunächst einige Definitionen aus der Systemsicherheit [Bishop04]

Subject

- ein Benutzer eines Systems oder
- ein Prozess, der von einem Benutzer ausgeführt wird

Object

Ressource, auf die Zugriff überwacht wird, z.B. Prozess oder Datei

Zugriffskontrollen II

Mandatory access control (MAC)

Abhängig von Systemrichtlinien darf ein Benutzer auf eine Ressource zugreifen.

Discretionary access control (DAC)

Die Identität des Benutzers bestimmt darüber, ob er auf eine Ressource zugreifen darf, festgelegt durch den Besitzer.

Role-based access control (RBAC)

Eine Rollenzuweisung ermöglicht einem Benutzer den Zugriff auf eine Ressource, kontrolliert vom System.

Zugriffskontrollen III

Principle of Least Privilege (PoLP)

- ein Subject kann benötigte Rechte vom System anfordern
- ein Subject darf höchstens die benötigten Rechte erhalten
- Modelle: BLP [Bell81]/[Bell05], Multi-Level Security (MLS)

Reference Monitor

- überwacht Zugriffe und stellt Einhaltung der Richtlinien sicher
- muss tamper-proof sein und darf nicht umgangen werden
- autorisiert Subjects via Datenbank, konfiguriert durch Admin

Security Kernel

Features:

- kann in Hardware, Firmware und Software integriert sein
- kryptografische Grundfunktionen
- Integritätsmechanismen (Modelle: Clark-Wilson, Biba)
- Zugriffsüberwachung
  - Autorisierung und Reference Monitor (s.o.)
  - Auditing für Nachvollziehen von Ereignissen
  - Non-executable Pages (NX)
- Schutz vor Buffer Overflows
  - Stack/Heap Protection
  - Address Space Layout Randomization (ASLR)

Security Kernel: Beispiel Linux Kernel Hardening

Für den Linux-Kernel stehen neben der Linux Security Modules (LSM)-Schnittstelle Patches bereit.

Atomicorp setzt ein und unterstützt:

- PaX (Stack/Heap Protection u.a., im Bundle mit grSecurity)
- grSecurity (implementiert u.a. RBAC, mit gradm verwaltet)

Vor dem Kompilieren muss auch die Toolchain gehärtet werden!
Es folgt ein Screenshot von der Linux-Kernel-Konfiguration, gepatcht mit PaX [PaX12]/[Vermeulen12]/[Vermeulen13].

Security Kernel: Beispiel Linux Kernel Hardening

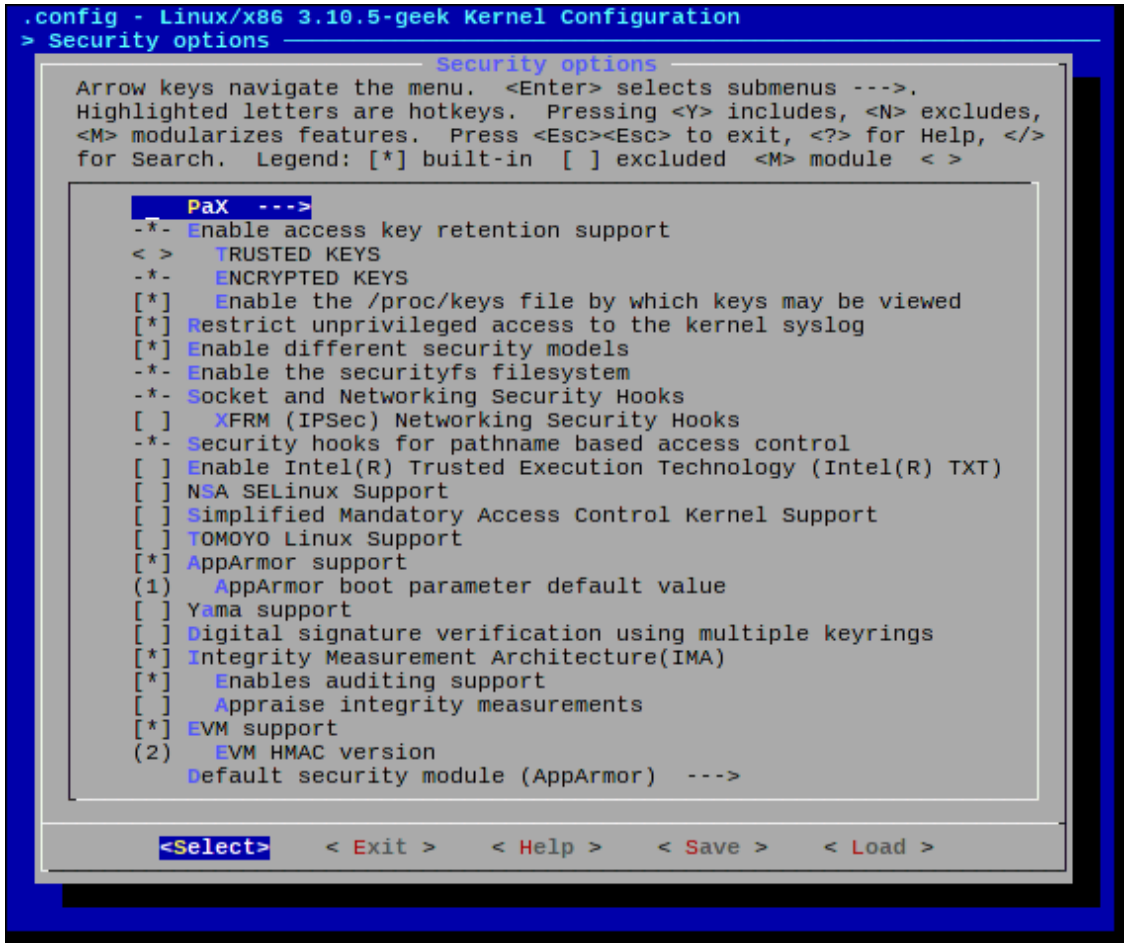


Figure : Linux Kernel-Security-Konfiguration

Common Criteria (CC)

Common Criteria for Information Technology Security Evaluation

- Sammlung von Sicherheitsnormen [CCRA12]
- Internationaler Standard (ISO/IEC 15408)
- Einstufung der Sicherheitseigenschaften von Systemen
- Die CC umfassen drei Teile:
  - Teil 1: Einführung und allgemeines Modell (Introduction and General Model)
  - Teil 2: Funktionale Sicherheitsanforderungen (Functional Requirements)
  - Teil 3: Anforderungen an die Vertrauenswürdigkeit (Assurance Requirements)

CC: Vorgehensweise für Zertifizierung

Die CC bezeichnen als Target Of Evaluation (TOE) das zu evaluierende Produkt bzw. System.
Für dieses werden schrittweise bestimmt:

- Schutzprofile (Protection Profiles, PP), d.h. Anforderungen an Funktionalität und Vertrauenswürdigkeit
- Security Target (ST), d.h. eines oder mehrere PPs
- Security Functional Requirements (SFRs), Teil des ST, individuelle Sicherheitsfunktionen des Produktes
- Security Assurance Requirements (SARs), Beschreibung der Maßnahmen zur Gewährleistung der Funktionen
- Evaluation Assurance Level (EAL), Einstufung des formalen Verifizierungsvorgangs des TOE

Evaluation Assurance Levels

Der jeweilige Evaluation Assurance Level eines Produktes oder Systems ermöglicht seinem Benutzer einen Abgleich mit seinen Anforderungen. Die sieben Stufen bedeuten jeweils:

1. funktionell getestet
2. strukturell getestet
3. methodisch getestet und überprüft
4. methodisch entwickelt, getestet und geprüft
5. semiformal entworfen und getestet
6. mit semiformal geprüftem Entwurf entwickelt und getestet
7. mit formal geprüftem Entwurf entwickelt und getestet

Separation Kernel: Eigenschaften

Idee [Rushby81]:

- Partitionierung und Isolierung von Hardware und Prozessen
  - Speicherbereiche, auf die je nur ein Prozess zugreifen darf
  - CPU-Rechenzeit in Zeitschlitze geteilt
  - Zugriffe auf weitere I/O-Ressourcen in Zeitschlitze geteilt
- spezielle Kanäle für die Kommunikation zwischen Prozessen
  - müssen konfiguriert und verwaltet werden
- Mikrokernel bieten wegen der geringeren Komplexität eine ideale Basis für eine effiziente Umsetzung (nach Rushby)

Ansatz:

- erweitere Mikrokernel um Partitionierung und geschützte IPC (Zugriffskontrollen, Reference Monitor)
- behalte Treiber und Dateisysteme weiterhin im User Mode

Separation Kernel: Definitionen (nach SKPP)

SKPP: Separation Kernel Protection Profile [IAD07]/[Levin10]

- Separation Kernel
  - Mechanismen in Hardware, Firmware und Software
  - Exekutive, die physikalische Ressourcen kontrolliert
  - abstrahiert verschiedene Typen von Ressourcen
  - lagert eine Teilmenge der Ressourcen aus
  - weist Ressourcen einzelnen Partitionen zu
  - kontrolliert den Informationsfluss zwischen Partitionen und zwischen Ressourcen
- Subjects: Teilmenge der ausgelagerten Ressourcen
- Objects: Ressourcen
- Fluss: Richtung durch Modus definiert
  - read: von Object zu Subject
  - write: von Subject zu Object

Separation Kernel: Partitionierung

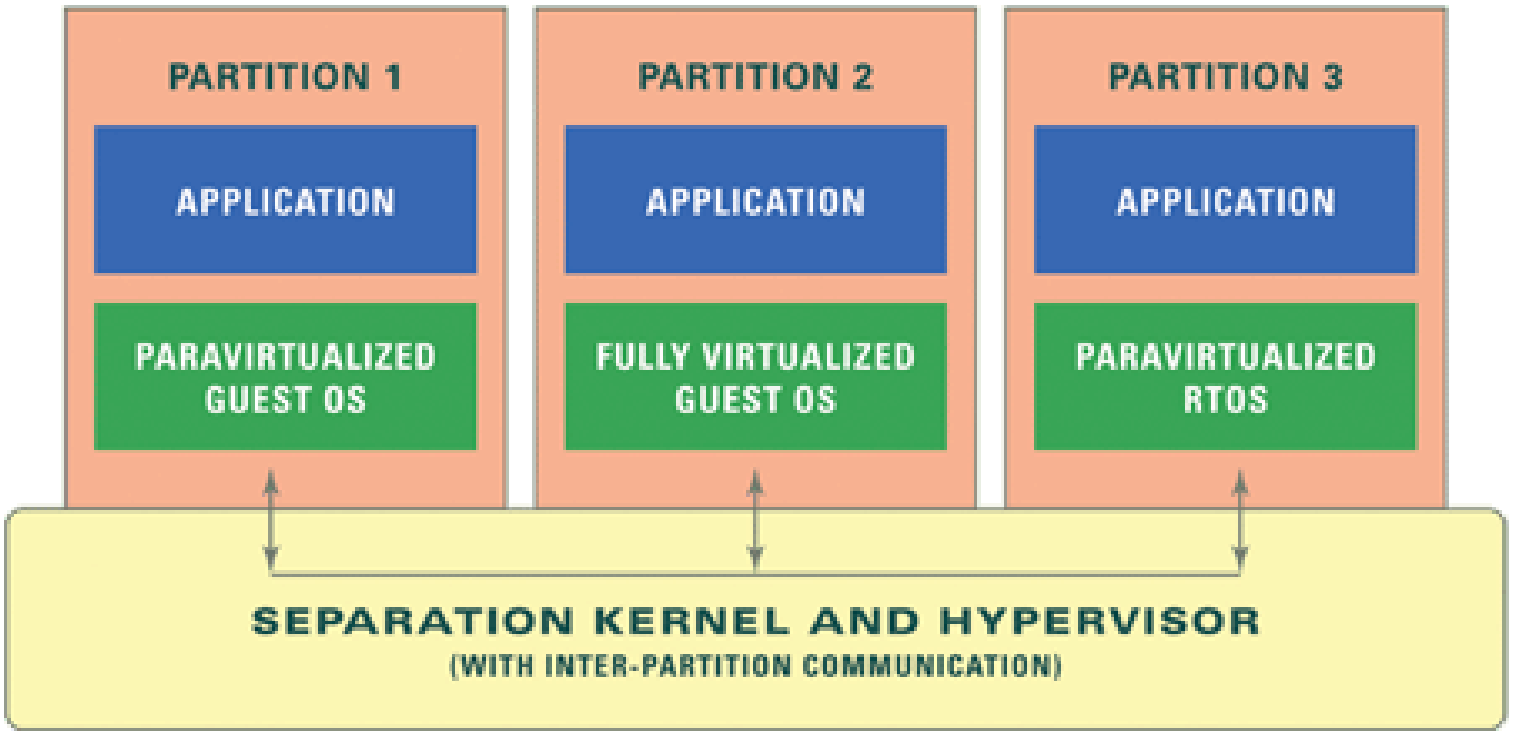


Figure : Separation Kernel: Partitionierung

Separation Kernel Protection Profile: Entwicklung

- Auftrag von der NSA an die MITRE Corporation und die Naval Postgraduate School (NPS)
- Ziel: Entwurf eines High Robustness Protection Profiles für Separation Kernels
- Basis: unveröffentlichtes Partitioning Kernel Protection Profile (PKPP)
- Entwickler: Open Group (McEvilley, Irvine, Nguyen, Levin)
- Erste Version: SKPPv0.621-2004-07-07
- beschreibt Produkte nach CC und Rushbys Konzept
- Pflichtenheft

Separation Kernel Protection Profile: Timeline

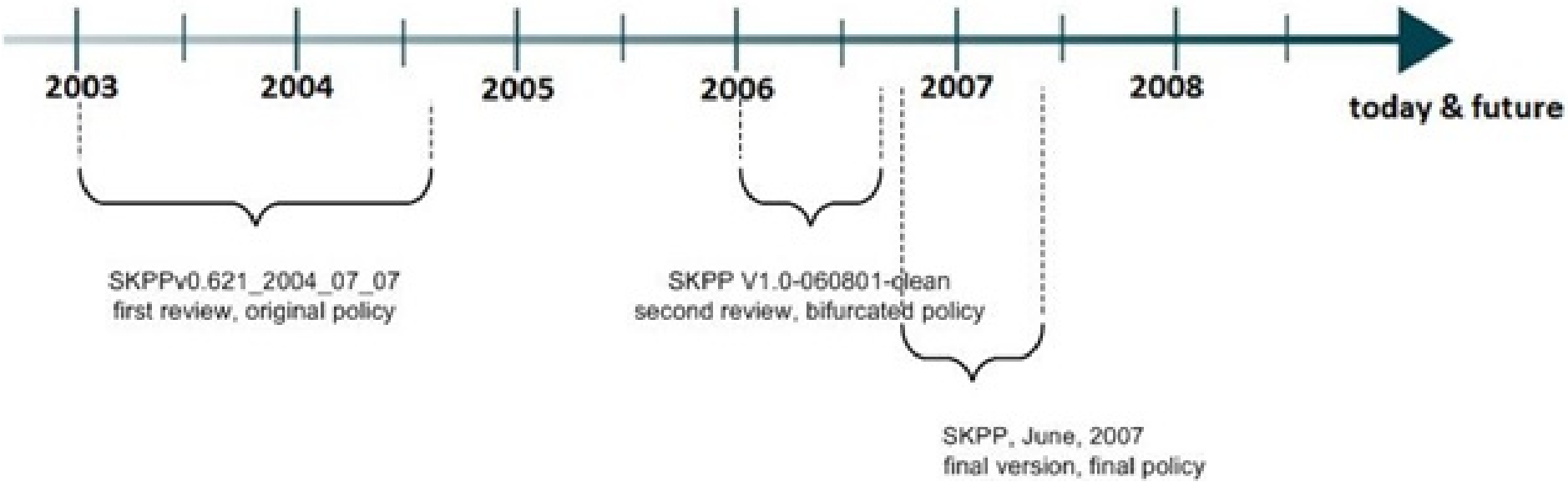


Figure : Entstehung des SKPP: Timeline

Separation Kernel Protection Profile: Anforderungen

Funktionale Sicherheitsanforderungen:

- Schutz von Ressourcen vor unautorisierten Zugriffen
- Trennung von internen Ressourcen
- Partitionierung und Isolierung von ausgelagerten Ressourcen
- Vermittlung der Informationsflüsse
- Audit über Erfüllung von Anforderungen und Richtlinien

Implementierung:

- zwei Matrizen für Zugriffskontrollen:
  - S2R (subject to resource)
  - P2P (partition to partition)
- nach Revision: sys.policy, Liste über aktive und inaktive Richtlinien

Separation Kernel Protection Profile: Policies

- Original Policy: Regel muss in *S2R* und *P2P* stehen

```
ALLOWED([s:subject, r:resource, m:mode]) =
      [s,r,m] ∈ S2R
      ∧
      [s.p,r.p,m] ∈ P2P
```

- Bifurcated Policy: Regel muss in *sys.policy* aktiv sein

```
ALLOWED([s:subject, r:resource, m:mode]) =
      S2R_p ∈ sys.policy →
            [s,r,m] ∈ S2R
      ∧
      P2P_p ∈ sys.policy →
            [s.p,r.p,m] ∈ P2P
```

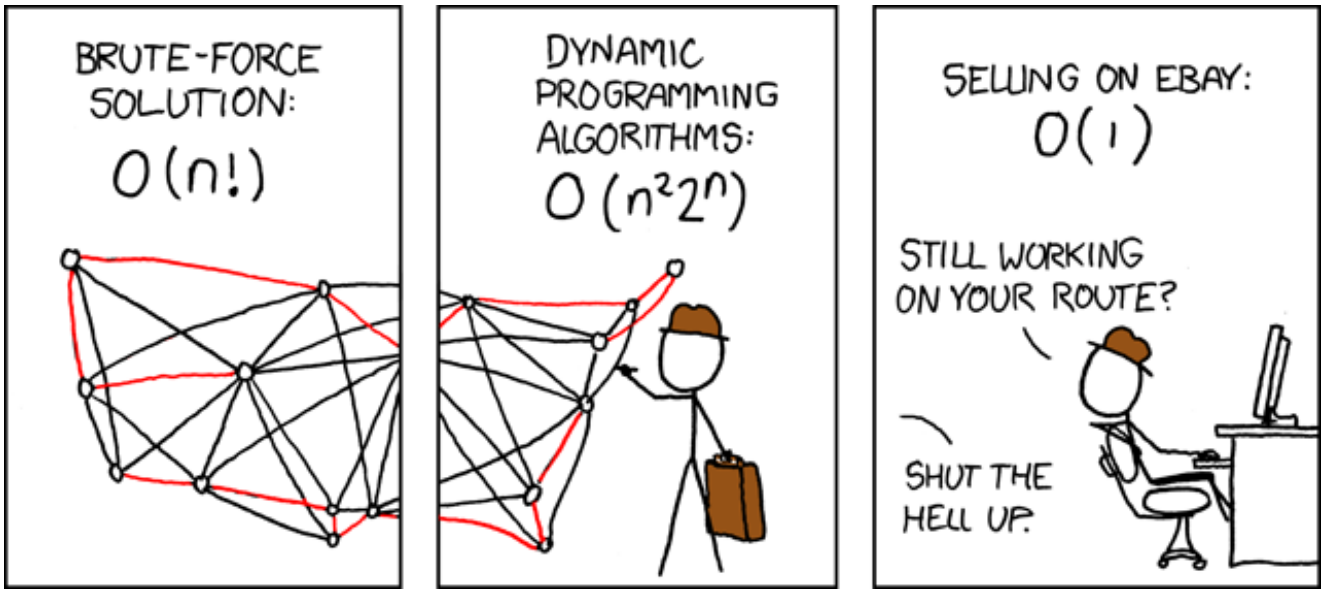
Separation Kernel Protection Profile: Policies

- Final Policy: $S2R$ überschreibt $P2P_p$ policy, wo nicht *null* steht

```
ALLOWED([s:subject, r:resource, m:mode]) =
      S2R_p ∈ sys.policy → (
              S2R(s,r).m = allow
              ∨
              (S2R(s,r).m = null
               ∧
               P2P(s.p,r.p).m = allow
              )
      )
      ∧
      P2P_p ∈ sys.policy →
              P2P(s.p,r.p).m = allow
```

Separation Kernel: Verifizierung

Problem: Der Code eines (monolithischen) Kernels ist komplex.



- formale Methoden
- Solver nach Reduktion, z.B. für SAT erforderlich
  - teuer da $SAT \in NP$ (Cook, Levin 1960er-1970er)

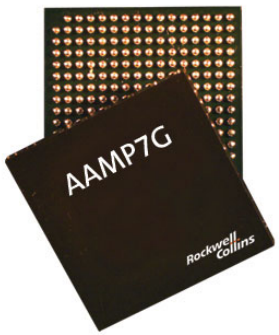
Formale Methoden

- Z-Notation (nach Zermelo-Fraenkel-Mengenlehre) [Spivey98]
  - Jean-Raymond Abrial, Oxford 1977
  - Tool: Z/EVES
  - Beispiele für Separation Kernel: [Velykis09], [Velykis10]
- Alloy (zuerst Nitpick, inspiriert von Z-Notation)
  - Daniel Jackson, MIT 1999
  - Alloy 4 2007
  - Tool (Solver): Alloy Analyzer
  - Beispiel für Separation Kernel: [Phelps08]
- SHADE (Secure High-Assurance Development Environment)
  - David Hardin et al, Rockwell Collins 2006 [Hardin06]
  - Beispiel für Separation Kernel: AAMP7G Prozessor

Anwendungsbereiche

Hardware

- Industrie: TPM (Trusted Platform Module)
- Akademische Forschung: Integration in CPU
- Militär: AAMP7G Prozessor



Software

- Virtuelle Maschinen: Hypervisor

Fragen?

Referenzen I


J. Rushby (1981) - Design and Verification of Secure Systems
http://www.csl.sri.com/papers/sosp81/sosp81.pdf


D.E. Bell, L.J. LaPadula (1973) - Secure Computer Systems: Mathematical Foundations
http://www.albany.edu/acc/courses/ia/classics/belllapadula1.pdf


D.E. Bell (2005) - Looking Back at the Bell-La Padula Model
http://www.acsac.org/2005/papers/Bell.pdf


M. Bishop (2004) - Introduction to Computer Security
http://nob.cs.ucdavis.edu/book/book-intro/


PaX Team (2012) - PaX - kernel self-protection
http://pax.grsecurity.net/docs/PaXTeam-H2HC12-PaX-kernel-self-protection.pdf


S. Vermeulen (2012) - Hardening Linux Kernel
http://dev.gentoo.org/~swift/docs/security_benchmarks/kernel.html


S. Vermeulen (2013) - Linux Sea
http://swift.siphos.be/linux_sea/linux_sea.pdf


CCRA (2012) - Common Criteria Publications
http://www.commoncriteriaportal.org/cc/

Referenzen II


J.M. Spivey (1998) - The Z Notation: A Reference Manual
http://spivey.oriel.ox.ac.uk/mike/zrm/zrm.pdf


A. Velykis (2009) - Formal Modelling of Separation Kernels
http://andrius.velykis.lt/publications/Velykis09.pdf


A. Velykis, L. Freitas (2010) - Formal Modelling of Separation Kernel Components
http://andrius.velykis.lt/publications/VelykisF10.pdf


D. Phelps, M. Auguston, T.E. Levin (2008) - Formal Models of a Least Privilege Separation Kernel in Alloy
http://www.cisr.us/downloads/papers/08paper_fm_alloy.pdf


D.S. Hardin, E.W. Smith, W.D. Young (2006) - A Robust Machine Code Proof Framework for Highly Secure Applications
http://www.ccs.neu.edu/home/pete/acl206/papers/hardin.pdf


Information Assurance Directorate (2007) - U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness
https://www.niap-ccevs.org/pp/pp_skpp_hr_v1.03.pdf


T.E. Levin, T.D. Nguyen, C.E. Irvine (2010) - Separation Kernel Protection Profile Revisited: Choices and Rationale
http://fm.csl.sri.com/LAW/2010/law2010-03-Levin-Nguyen-Irvine.pdf