Vertrauen in moderne IT-Infrastrukturen

Anforderungen an die Betriebssysteme-Konstruktion

Claudia Eckert, F. Stumpf, C. Schneider

Fraunhofer Institut SIT, Darmstadt/München
TU München

Betriebssysteme Workshop, München, 4.12. 2008

Fraunhofer Institut Sichere Informations-Technologie

Gliederung

1. Einführung
2. Existierende Ansätze (Auswahl)
3. Einige Probleme, Fragestellungen
4. Diskussionspunkte

1. Einführung

Situation

Zunehmender Einsatz von IT-Systemen

- in Mission-Critical, Hochsicherheits-Umgebungen
- in eingebetteten Systemen und
- in mobilen Umgebungen
- ubiqutitäre Zugriffe auf schützenswerte Güter

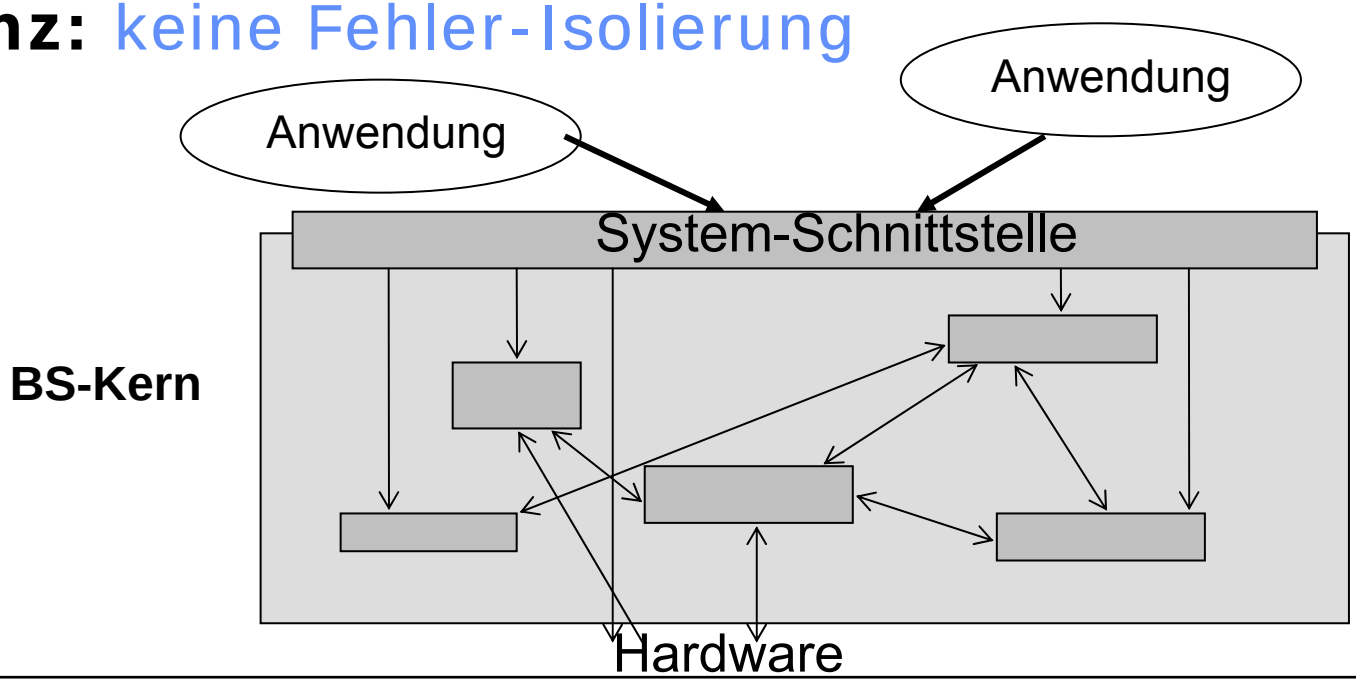
Gleichzeitig

- Übergang von speziellen, gut beherrschbaren Mikrocontrollern mit dedizierter Software zu
- komplexen IT-Systemen unter Nutzung von
- Standard-Betriebssystemen

Standard-Betriebssysteme

Monolitisch (Unix, Linux, Windows XP...)

- komplexer Systemkern, z.B. Linux: >2.5 Mio LoC, Analysen: 6 Bugs/1T LoC
- **Kern**: tausende von gebundenen Prozeduren in einem Adressraum, Ausführung im privilegierten Modus
- **Konsequenz:** keine Fehler-Isolierung



Hardware-Architekturen

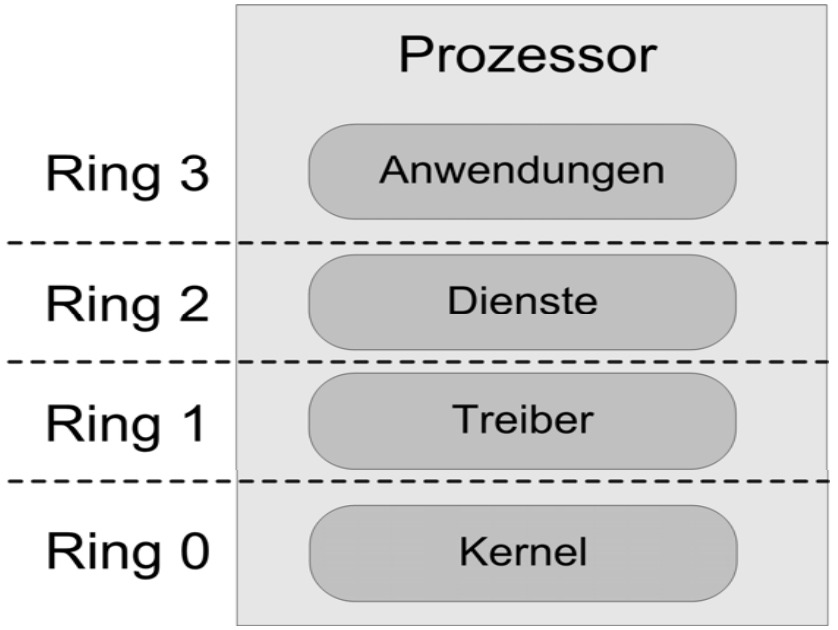
- einige CPU-Architekturen unterstützen differenzierte Schutzdomänen durch Schutzringe
- d.h. Architekturen bieten unterschiedliche Sicherheitslevel

Schutzringe:

- Unterschiedliche Privilegien
- Hardware-basierte Kontrollen

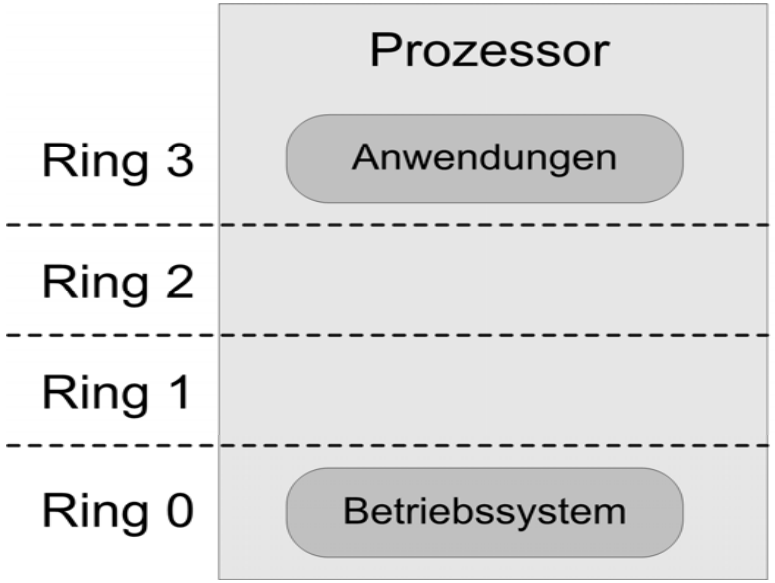
Nutzung: z.B.

- Treiber-Code in Ring 1
- Treiber besitzen
  keine Kernel-Privilegien



Problem:

- Lücke im Zusammenspiel von Hardware u. Betriebssystem
  idR nur 2 Schutzringe genutzt



Monolithische BS:

- alle BS-Komponenten in Ring 0
- können Speicherbereiche aller
  Dienste lesen/schreiben
- Problem Gerätetreiber,
  sehr kritisch, hohe Anzahl von Fehlern/Schwachstellen

Konsequenz:

- Integritätsverletzung von BS-Komponenten ist einfach!

Ziele bei Betriebssystem-Kostruktion:

- bessere Isolierung, Begrenzung des Schadensausmaßes
- erhöhte Robustheit, Verlässlichkeit

Wichtige Lösungsansätze:

1. Härten von BS-Komponenten
2. Virtualisierung
3. Mikrokernbasierte Systeme
4. Trusted Computing

Gliederung

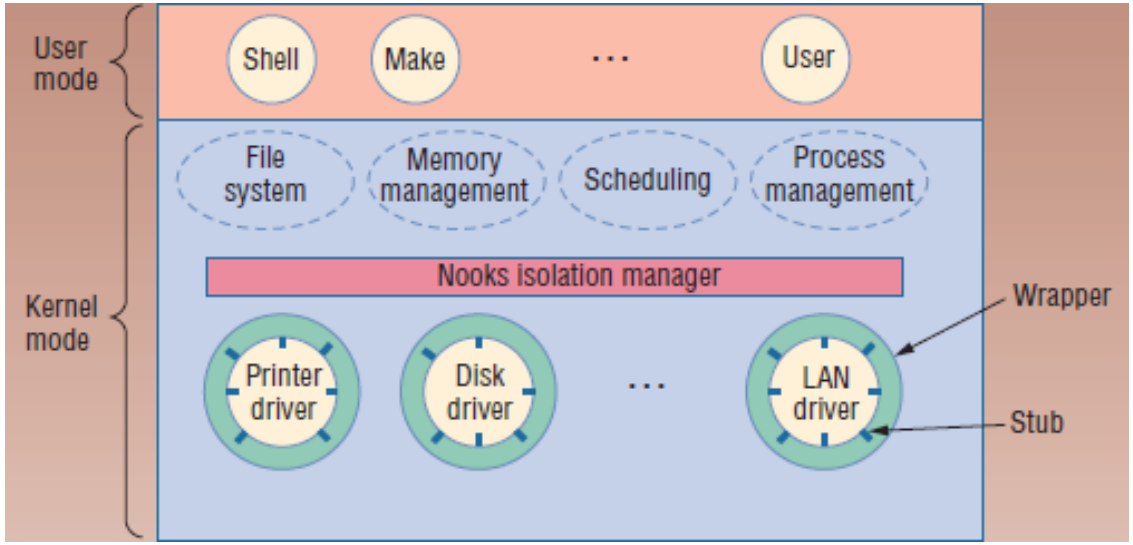
1. Einführung
2. **Existierende Ansätze**
3. Einige Probleme, Fragestellungen
4. Diskussionspunkte

1. Härten von BS-Komponenten

Ziel: Höhere Robustheit, geringe BS-Modifikationen

Ansätze u.a.

- Wrapper: z. B. Parameterprüfung
- Software-Fault-Isolation:
  Safe code Erweiterung



Beispiel: Nooks

Ziel: Schutz des Kerns

Ansatz

- Sandbox für Treiber
- Wrappen von Aufrufen in Treiber/vom Treiber
- Wiederherstellung des BS-Zustands nach Treiber-Crash

Probleme: u.a. unsicherer Treiber-Code, manuelle Wrapper

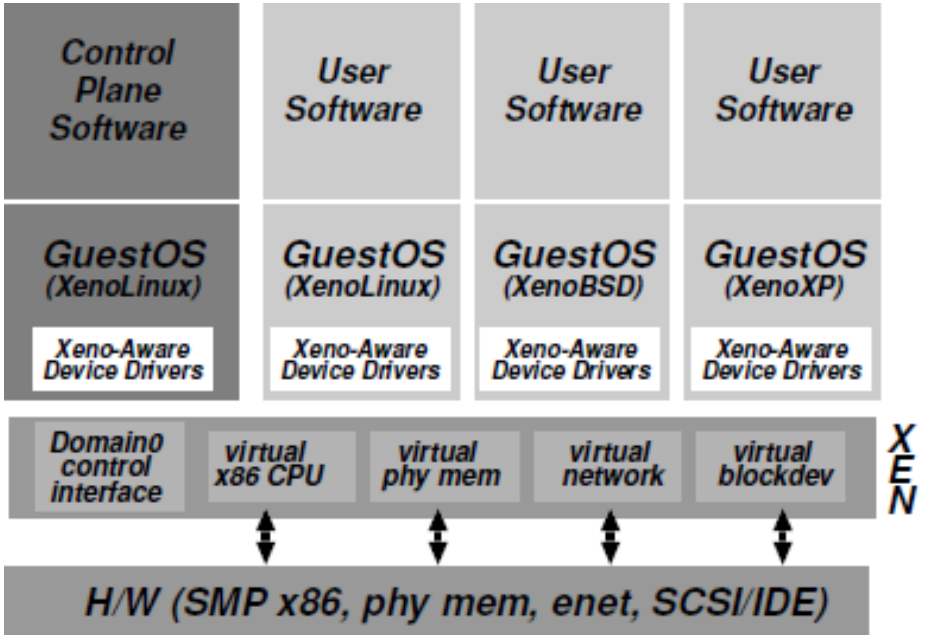
2. Virtualisierung

VMM: Virtual Machine Monitor:

- Kontrollprogramm, direkt auf der Hardware, unterhalb von BS
- Verschiedenen BS in eigener virtueller Maschine (VM)
- VM realisieren Fault Isolation

Beispiel Xen-Hypervisor:

- Effizient
- Explizite Kommunikation zw. VMs über Shared Pages (erfordert Paravirtualisierung, Hypercalls)

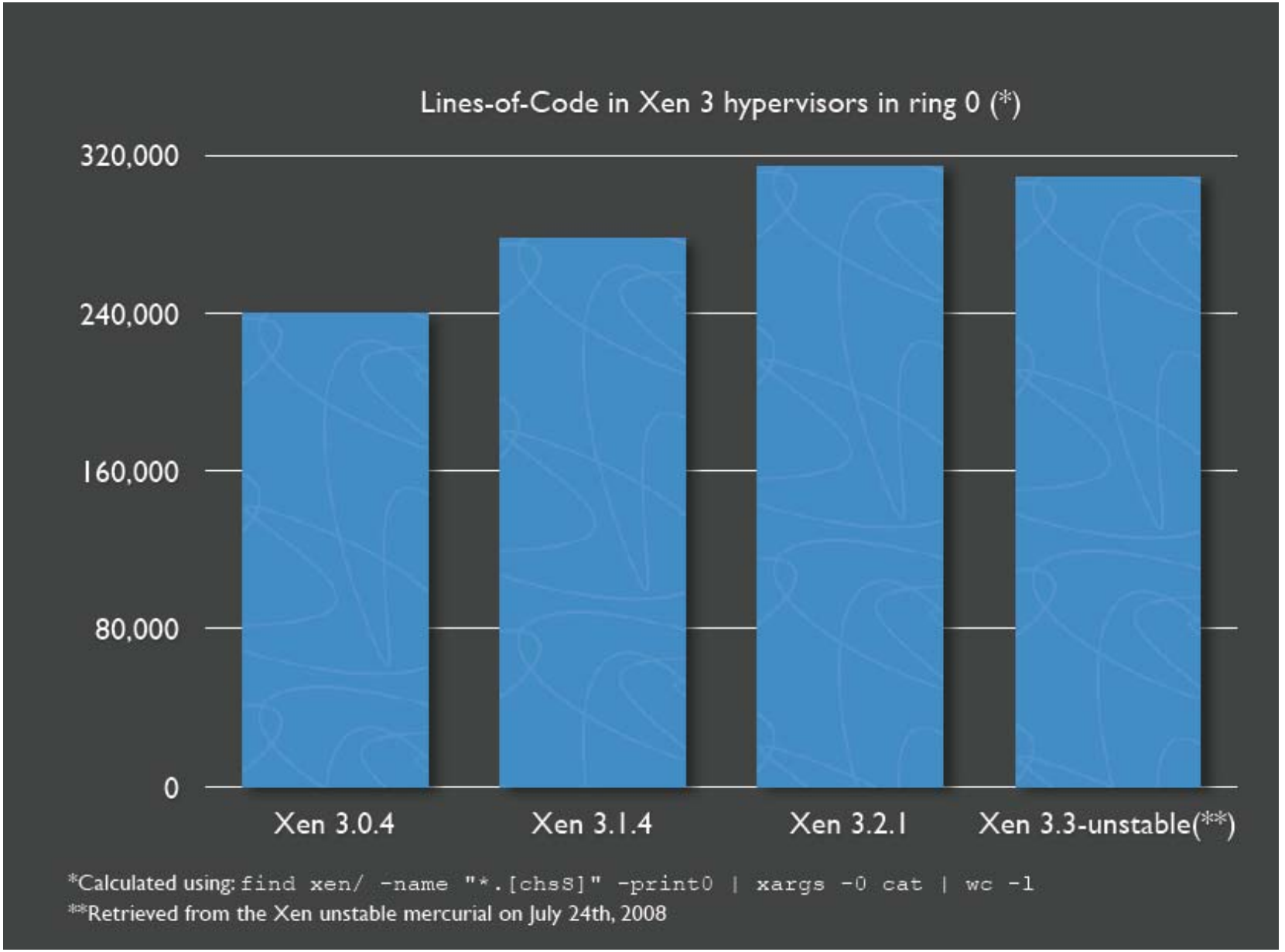


Probleme: u.a.

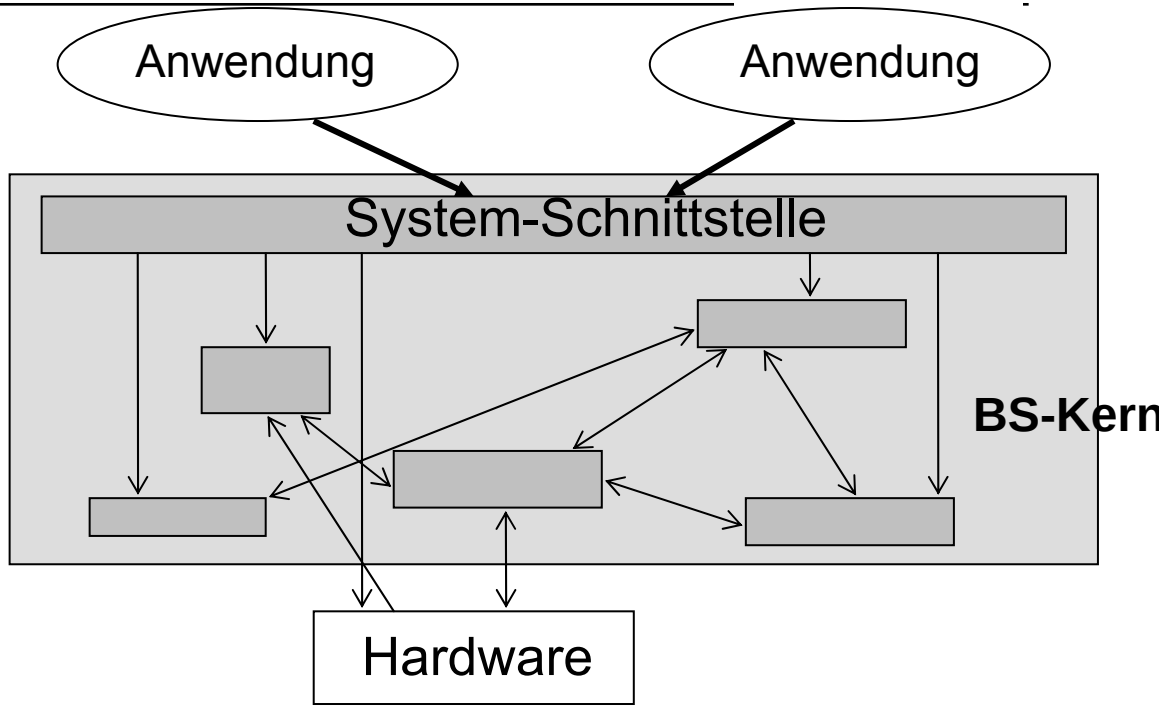
- Beherrschbarkeit: Hypervisor werden immer komplexer

Problem: Komplexität!

- Fehleranfällig,
- Ausführung in privilegierten Modus: Ring 0



3. Mikrokern

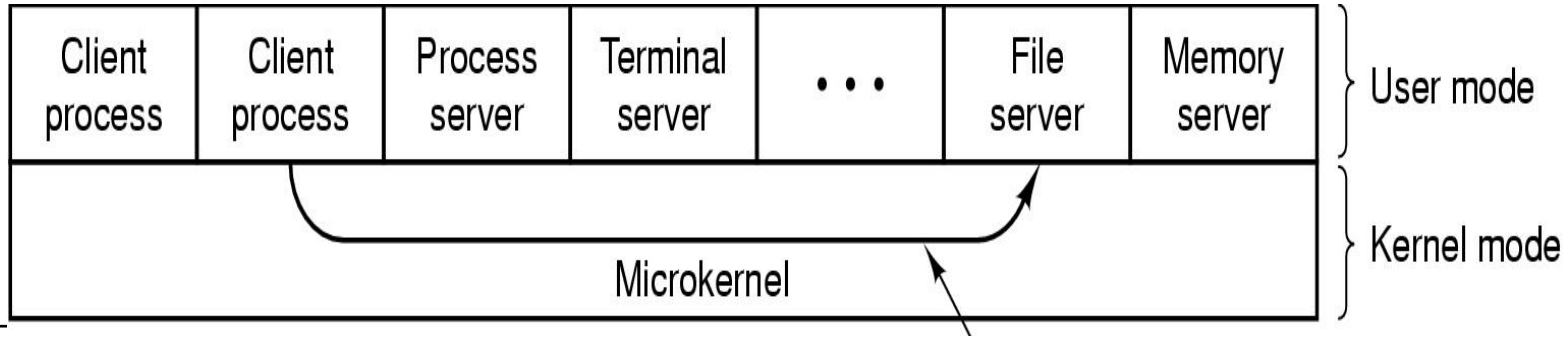


Monolitisch (Unix, Linux,...)

- komplexer Systemkern

Mikrokernbasiert

- klein, beherrschbar, verifizierbar, TCB
- Eignung für embedded Umgebungen

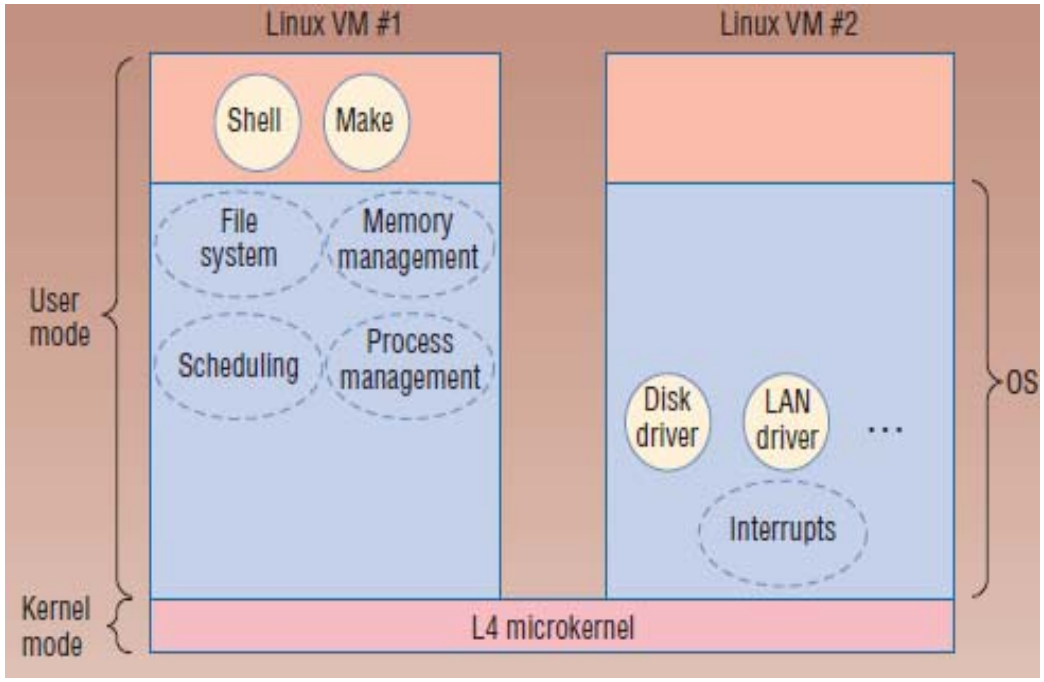


3. Mikrokernbasierte Systeme (Forts.)

Beispiele: L4, Minix, MACH, ...

Ansatz:

- minimales Dienstespektrum:
  - Threads & Adressräume
  - IPC



Beispiel L4

- Kombination mit VM-Konzept
- z.B. Isolierung von Treibern in VM

  Treiber-Crash beendet nur die VM, Rest des BS bleibt in Takt

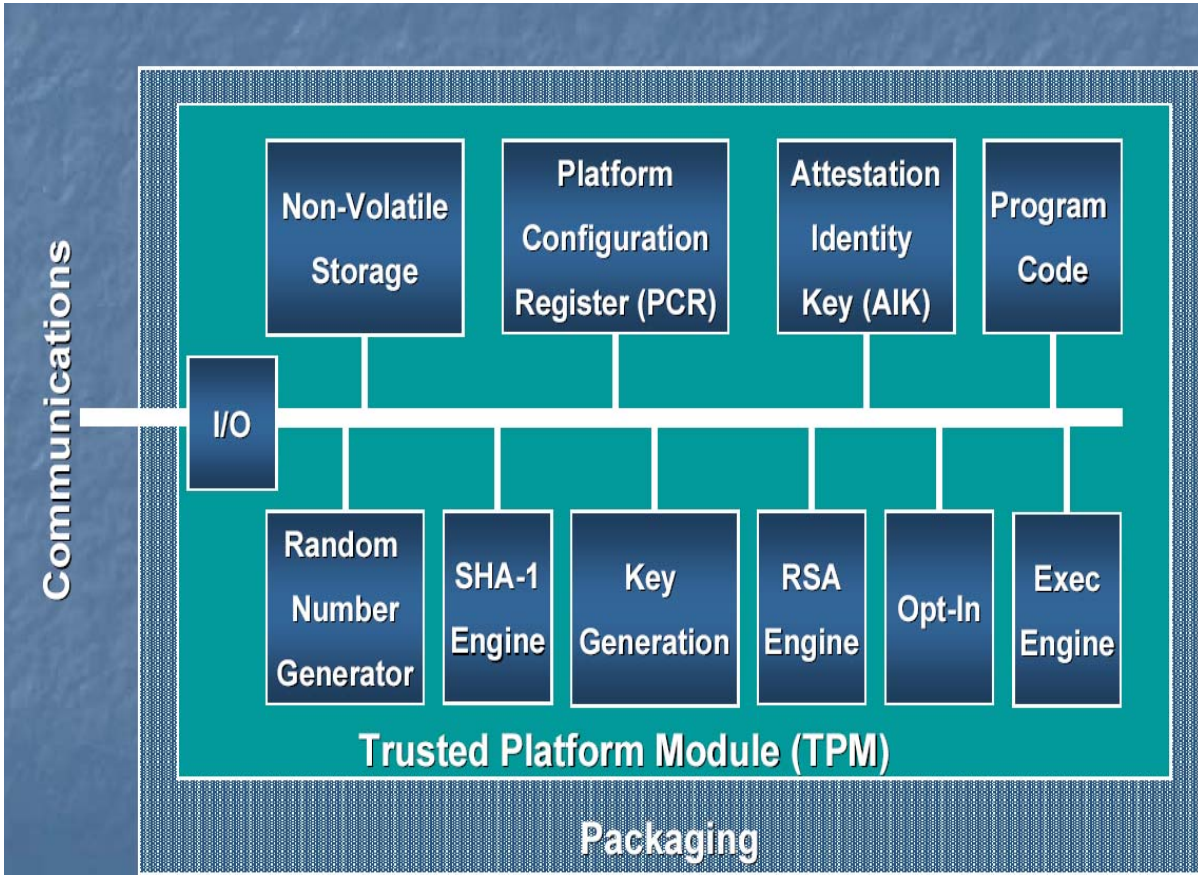
Problem:

- Interaktion über VM-Grenzen: Performanzverlust?

4. Trusted Computing

Ansatz:

- Integrität von BS-Komponenten sicherstellen
- Sicherheits-Features:
  - „Sicheres“ Booten,
  - Sealing, Attestierung, ..
  - Sicherer Speicher



Problem: u.a.

- Integrität von einzelnen Prozessen sicherstellen:
  - Prozesse an Plattformkonfiguration binden (sealing)?
  - Aber: unpraktikabel bei monolithischen Strukturen

Gliederung

1. Einführung
2. Existierende Ansätze
3. **Einige Probleme, Fragestellungen**
4. Diskussionspunkte

Einige Probleme/Fragestellungen

Vermischung: Mikrokern und Virtualisierung

- keine klare konzeptuelle Abgrenzung
- systematische Evaluierung notwendig?!

Problembereiche: u.a.

1. Gemeinsame Ressourcennutzung
2. Grenzen der Software-Virtualisierung
3. Vertrauenswürdigkeit von Hypervisor-Software
4. Komplexität von VMMs/Kernen
5. Trusted Computing Probleme

1. Gemeinsame Ressourcennutzung

Problem: Kommunikation zwischen isolierten VMs

- indirekt, nicht kontrolliert:
- Konsequenz: nicht kontrollierte, verdeckte Kanäle!
  z.B. Kommunikation über Netzwerkgeräte oder IPC
- Aber: Kontrollierter Datenaustausch häufig gewünscht

Erforderlich:

- Kommunikation zw. sensitiven/nicht-sensitiven Bereichen:
  Ziel: Informationsflüsse kontrollieren und filtern

Lösungsansätze?

- MAC-Policies im Hypervisor zur Informationsflußkontrolle?
- Trusted Channels von z.B. Browser zur VM?

2. Grenzen der Software-Virtualisierung

Beispiel:

- Framebuffer von Grafikkarte nicht virtualisierbar
  - einer VM, idR der Dom0 zugewiesen (privilegierte VM)
  - Ausführung z.B. des X-Servers in Dom0
- **Konsequenz:**
  - VMs nutzen für ihre Ausgabe X-Server einer anderen VM
  - z.B. Trusted Signer muss fremde VM nutzen:
    welche Daten werden dann wirklich angezeigt?

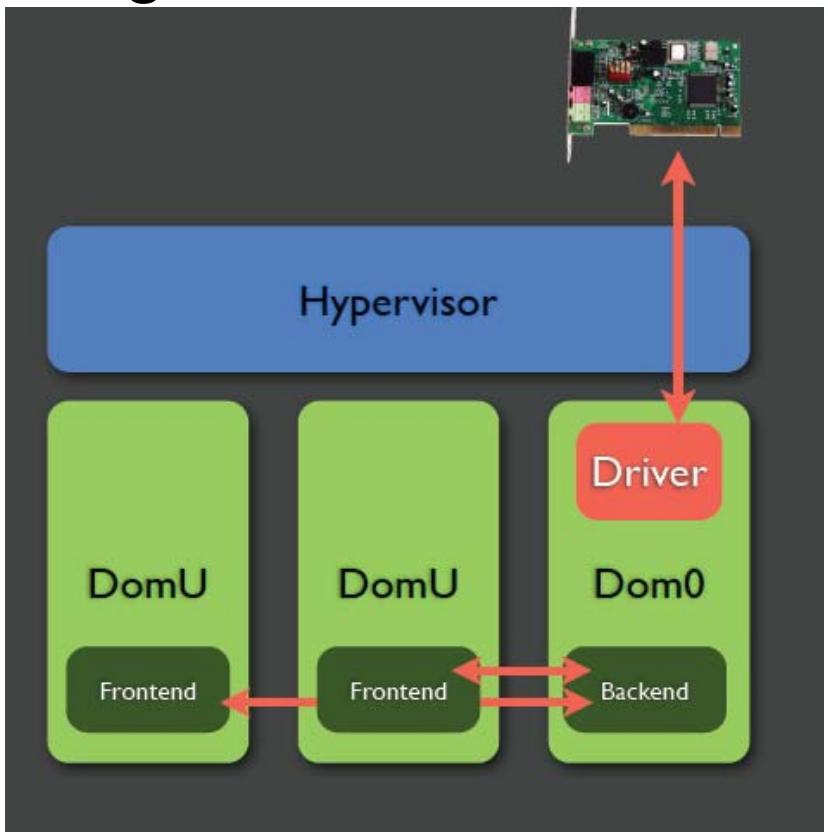
Lösungsansätze:

- Virtualisierung (der Graphikkarte) in Hardware
  - Jede VM bekommt exklusive Zugriffe
  - Aufbau eines Trusted Paths zwischen X-Server und VM

2. Grenzen der Software-Virtualisierung

Beispiel: Treiber

- Pro VM ein Frontend-Treiber
- kommuniziert mit Backend-Treiber
- Backend-Treiber in VMM oder in Management Domäne (dom 0)
- Backend: Multiplexen der Treiberzugriffe



Problem:

- Treiber mit hohen Privilegien!

Lösungsansätze:

- Virtualisierung in Hardware? Neue Hardware-Architekturen?
  - Kontextwechsel für VMs, virtuelle Interrupts, ..,

2. Grenzen der Software-Virtualisierung

Beispiel: DMA

Problem:

- DMA umgeht Speicherschutz
- Sprung an beliebige Adressbereiche ist möglich
  Isolation bei VM wird aufgehoben

Lösungsansätze:

- Spezielle Hardware: I/O MMU:
  Schutz physikalischer Seiten vor DMA-Geräten

Problem:

- von der Software noch nicht unterstützt

3. Vertrauenswürdigkeit von Hypervisor-Software

Neue Angriffe durch Ausnutzung der VMM-Technik!

Problem:

- Installation eines bösartigen Hypervisor
  durch z.B. Virtual Machine based Rootkits (VMBR)

Ansatz: z.B Einschleusen des VMVR während des Bootens

Konsequenz:

- Kompromittierter Hypervisor kontrolliert alle VMs
- Kann seine Aktivitäten verstecken, z.B. vor IDS etc
- Kann weitere Malware-Dienste in einer VM installieren

Lösungsansätze:

- Erkennung: Sichere Hardware ‚unterhalb' des VMBR: TC
- Booten von sicherem Device, Secure VM (TC-basiert?)

4. Komplexität von VMMs /Kernels

- Minimale TCB versus
- effizientes Ausführungen in komplexen Security Kernels

Probleme u.a.

Designkriterien: welche Komponenten in den Kernel!

- Einordnung von Sicherheitsprotokollen (IPSEC, TLS, Attestation)
  - Protokollstack in Kernel?
    komplex, eine privilegierte Domäne (kennt alle Schlüssel)
  - Protokollstack in den User-Mode/VM?

Lösung? Lessons learned aus Mikrokern-Entwicklung?

5. Trusted Computing

Probleme u.a.

- TC-Infrastruktur fehlt
  - Fehlende Hersteller EK-Credentials
  - Keine Privacy-CAs verfügbar
- Für Embedded ungeeignet:
  Spezifikation ist zu komplex
- Kein Schutz gegen Runtime-Angriffe (Buffer Overflow etc.)

Lösungsansätze:

- Architekturen? Harvard-artig?
  - Trennung: Daten- und Codebereich
- dedizierte trusted/secure Hardware-Elements für Embedded?

Anforderungen an BS-Konstruktion: Let‘s discuss!

These 1: Kernel-based VMs sind vielversprechend

- **offen:** Beherrschen der Dynamik (inkrementelle Beweise?)

These 2: Verlagerung von Virtualisierung in die Hardware ist notwendig:

- **offen**: Design geeigneter Hardware-Architekturen

These 3: Hardware/Software Co-Design ist notwendig

- **offen:** Modell-getriebene, durchgängige Konstruktionsprozesse?

These 4: Vertrauenswürdige VMM ist erforderlich

- **offen:** Beyond TPM, welche Secure Elements für Embedded

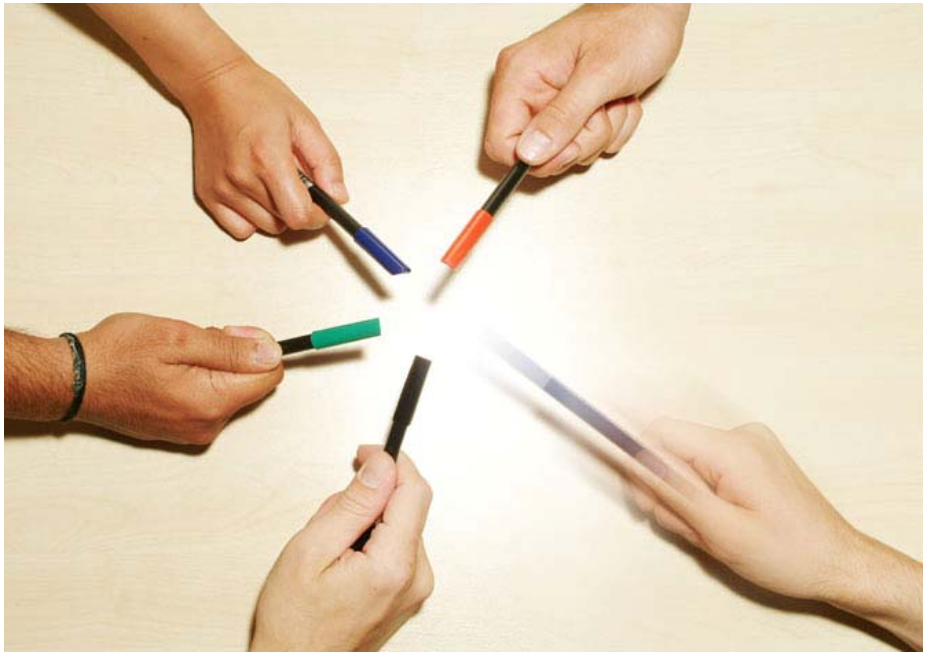
Trusted Plattform: µ-Kern + secure VMM + TC + HDW-Support ?

Vielen Dank für Ihre Aufmerksamkeit

Fraunhofer Institute for Secure Information Technology SIT
Prof. Dr. Claudia Eckert

Rheinstraße 75
64295 Darmstadt, Germany
Phone: +49-6151-869-358

mail: eckert@sit.fraunhofer.de
www: http://www.sit.fraunhofer.de



Fraunhofer SIT Institut Sichere Informations-Technologie