



Technische
Universität
Braunschweig

Institut für Betriebssysteme
und Rechnerverbund



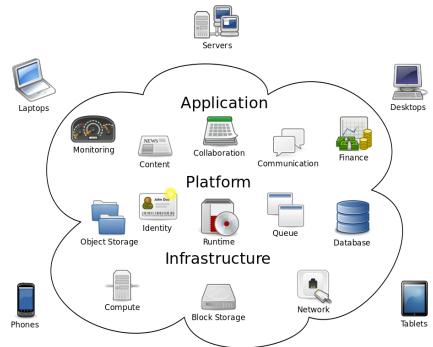
SecureBlue++

Seminar Informatik WS 2016/2017 Verteilte Systeme:
Secure Trusted Execution for Resilient Distributed Systems

Stephan Mielke, 14. Dezember 2016

Motivation

- ☺ Cloud-Computing fast überall
- ☺ Auslagern der Hardwarepflege
- ☺ Billige Rechenleistung
- ☺ Leistung On-Demand



Cloud-Computing Übersicht [4]

Motivation

- 😊 Cloud-Computing fast überall
- 😊 Auslagern der Hardwarepflege
- 😊 Billige Rechenleistung
- 😊 Leistung On-Demand
- 😞 Daten werden manipuliert
- 😞 Systemverhalten geändert
- 😞 Anwendungen werden abgehört



NSA Logo [3]

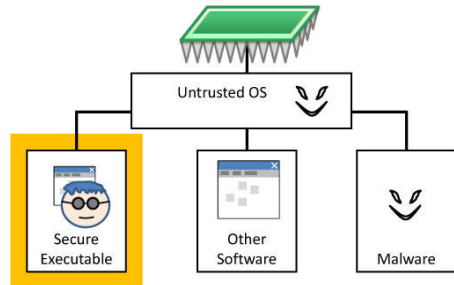
Motivation

Was wir wollen:

- Transparent für die Anwendung und das OS
- Anwendungen als Blackbox
- Isolierte Anwendungen
- Keinem Vertrauen

Was wir nicht beachten:

- Sicherheitslücken
- Programmierfehler
- Side-Channel Attacken
- Denial-of-Service Attacken



IBM Research Report [1]

Überblick

- **SecureBlue++**
 - Funktionsweise
 - System Calls
 - Anwendung
 - Multithreading
 - Virtuelle Maschinen
 - Benchmark
- **Intel SGX**
 - Vergleich
- **Zusammenfassung**

SecureBlue++

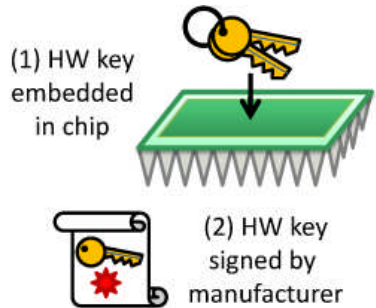
- Seit 2011 in Entwicklung
- Bisher keine reale Implementierung
- Sicheres, isoliertes Ausführen von Anwendungen
- Befehlssatzerweiterung für die POWER Architektur
 - ⇒ Neue Register, Instruktionen
 - ⇒ Geändertes Verhalten
 - ⇒ Einzigartiger Schlüssel
 - ⇒ Transparent für die Anwendung
 - ⇒ Nahezu transparent für das OS



IBM Logo [6]

SecureBlue++ – Neue Register und ROMs

- Secure Executable ID Save/Restore
 - *SEIDSR*
- Secure Executable ID
 - *SEID*
- Eigener Private/Public Key



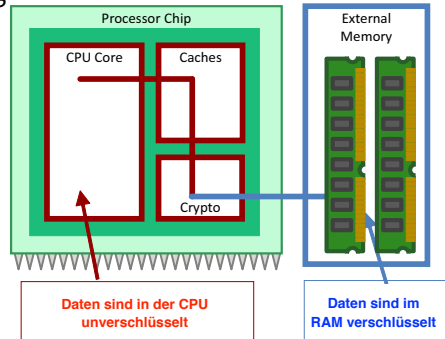
Schlüssel [1]

SecureBlue++ – Interne Datenstrukturen

- Secure Executable Table (*SET*) – global
 - *SEID*
 - Hash-Wert der Metadaten (*MRMT*, Zeiger auf *TRL*, Signal Handler ...)
 - Zeiger auf Metadaten
- Memory Region Mapping Table (*MRMT*) – lokal
 - AES-Schlüssel
 - Größe und Start der Region im Speicher
- Protected Memory Table (*PMT*) – global
 - *MRID*
 - *SEID* des Erstellers
 - Größe und Start der Region im Speicher
 - AES-Schlüssel
- Thread Restore List (*TRL*) – lokal

SecureBlue++ – Verhalten

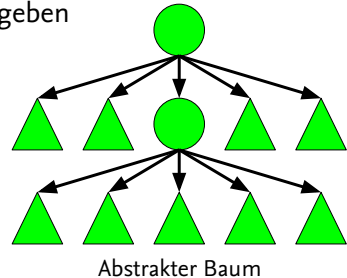
- RAM-Verschlüsselung
 - Anwendungseigener AES-Schlüssel
- Cache-Line-Schutz und Validierung
 - Integrity Tree Validierung
 - MRID Labeling
- Register-Schutz
 - Bei jedem Interrupt
 - CPU sichert Register in TRL
 - Nur SEID-Register bleibt
 - Nicht bei sesc-Instruktion



Grundidee [1]

SecureBlue++ – Integrity Tree

- Menge von Hash-Werten des Speichers
- Knoten werden durch Elternknoten geschützt
- Dient zum Schutz des Speichers vor Manipulationen
- Initiale Zustand wird esm in Metadata übergeben



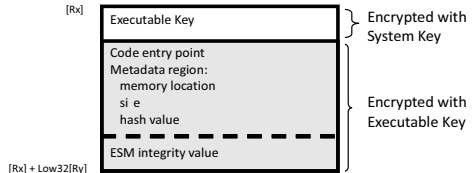
SecureBlue++ – Neue Instruktionen

- **esm**
 - Erstellt die Verwaltungsdaten
 - Startet die Anwendung
 - Privilegiert
- **restorecontext**
 - Stellt die Register wieder her
 - Privilegiert
- **deletecontext**
 - Gibt die Ressourcen frei
 - Privilegiert
- **sesc**
 - Führt System Calls aus



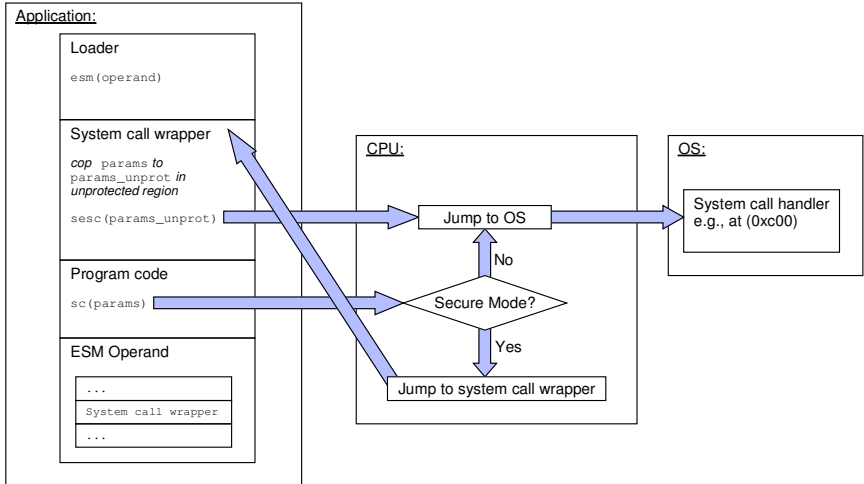
Rx = Register containing memory location of ESM operand
 Ry = Register with si e of ESM operand in low 32 bits;
 version # in high 32 bits

ESM instruction operand in memory:



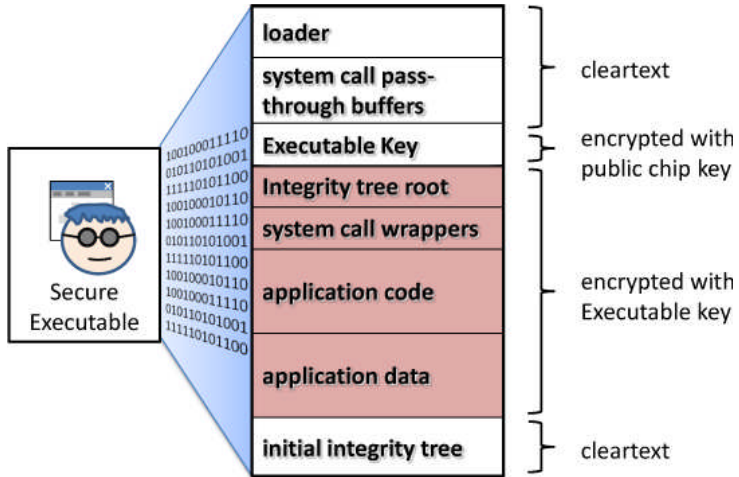
esm-Instruktion [1]

SecureBlue++ – System Calls



System Calls[1]

SecureBlue++ – Aufbau der ELF (Anwendung)



Aufbau[1]

SecureBlue++ – Multithreading

User Threads:

- Laufen auf Benutzerebene
- Eigener Scheduler im Programm ruft `restorecontext` auf

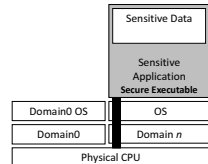
Kernel Threads:

- Kernel Verwaltet die Threads
- Viele Zustände für eine Anwendung
 - Einen pro Thread
 - Gespeichert in *TRL*

SecureBlue++ – Virtuelle Maschinen

Hardware VM:

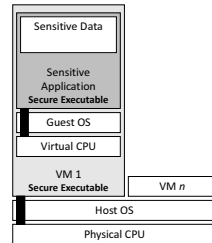
- Bevorzugt, weniger Leistungseinbußen
- Gleiches Verhalten wie ohne VM
- Domain 0 ruft restorecontext auf



Hardware VM[1]

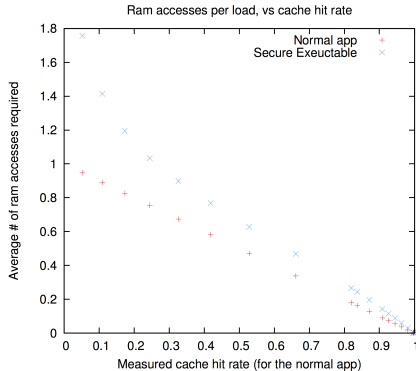
Software VM:

- Benötigt Anpassungen
- Gast OS als sichere Anwendung
- Anwendung als virtuelle sichere Anwendung

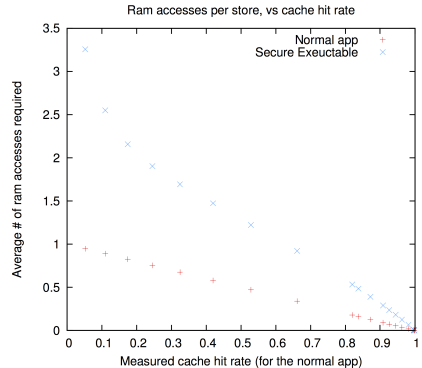


Software VM[1]

SecureBlue++ – Benchmark



Lesen [1]



Schreiben [1]

- Sequenz von 10^6 Adressen als Exponentialverteilung



Nur Simulation mit Cache Line-Validierung

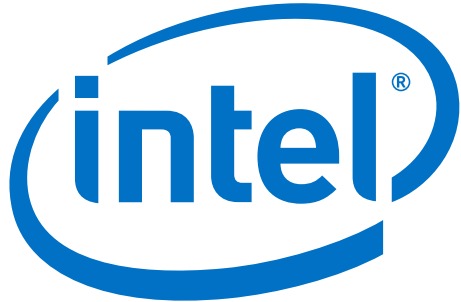


Keine Verschlüsselung



Intel SGX

- Befehlssatzerweiterung von Intel
- Ermöglicht sichere und isolierte Ausführung
- In Skylake CPUs enthalten
- Limitiert auf 128MB RAM



Intel Logo [7]

Intel SGX – Vergleich

	SecureBlue++	Intel SGX [5]
Feinheit des Schutzes	Prozess	Virtuelle Speicherregion
OS in der TCB	N	N
Limitiert die Anwendung	N	N
RAM Validierung hardwarebeschleunigt	N	J
Dynamisch wachsender Speicher	J	N
Anwendung ist Verschlüsselt	J	N
Geheimnisse nur im speziellen Bereichen	N	J
Portabel	N	J

Angelehnt an [2]

Zusammenfassung

- 😊 SecureBlue++ verspricht viel
- 😊 Isoliert Anwendungen
- 😊 Transparent für die Anwendung
- 😊 Geringe *TBC*

Zusammenfassung

- 😊 SecureBlue++ verspricht viel
- 😊 Isoliert Anwendungen
- 😊 Transparent für die Anwendung
- 😊 Geringe *TBC*
 - POWER Architektur
 - Möglicherweise in den POWER9 CPUs enthalten
- 😞 Bindung an eine CPU macht Cloud Computing nahezu unmöglich
- 😞 Sehr wenig verfügbare ungenaue Informationen
- 😞 Lässt sich selbst für Malware missbrauchen

Danke

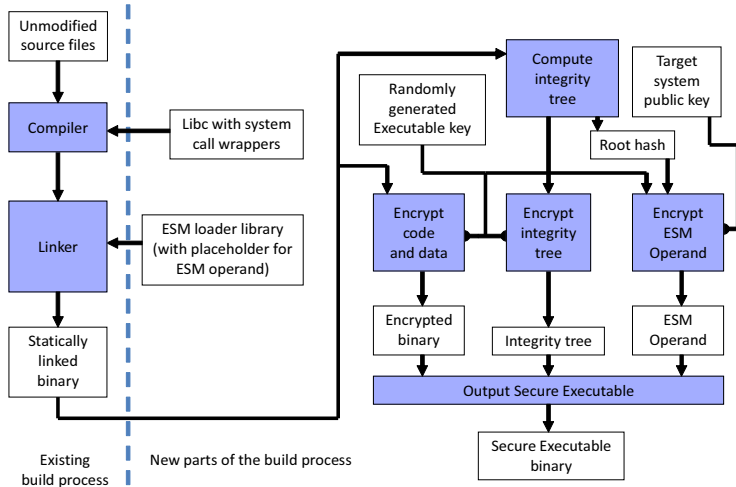
Vielen Dank für die Aufmerksamkeit und das Interesse!

Backup – System Calls

System Calls müssen selbst implementiert werden:

- `signal`
 - OS kann nicht das *PC*-Register ändern
 - Eigener Signal Handler in den Metadaten
- `fork`
 - Speicherbereich wird kopiert
 - Eigentümer in *SET* wird geändert
- `clone`
 - Eintrag in *TRL*
 - Einträge dem Scheduler zur Verfügung gestellt
- `exit`
 - Programm wird in eine Endlosschleife versetzt

Backup – Erstellen der Anwendung



Erstellen[1]

Quellen I



R. Boivie and P. Williams.

Secureblue++: Cpu support for secure executables.
Technical report.



D. Evtushkin, J. Elwell, M. Ozsoy, D. Ponomarev, N. A. Ghazaleh,
and R. Riley.

Iso-x: A flexible architecture for hardware-managed isolated
execution.





*In 2014 47th Annual IEEE/ACM International Symposium on
Microarchitecture, pages 190–202. IEEE, 2014.*



U. Government.

The seal of the u.s. national security agency, 1966.

Quellen II

-  S. Johnston.
Diagram showing overview of cloud computing, 2009.
-  F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi,
V. Shanbhogue, and U. R. Savagaonkar.
Innovative instructions and software model for isolated execution.
In *HASP@ ISCA*, page 10, 2013.
-  P. Rand.
Ibm logo, 1972.
-  Unbekannt.
Ibm logo, 2012.